

# MOCK·ING·BOARD



**User's Guide**

# M O C K I N G B O A R D

## PREFACE

### 1981

In 1981 Sweet Micro Systems releases a sound card for the Apple II. It was available in two major revisions, the RST, the “Sound” series, having sound and or speech options, or both, the later revision, officially named MOCKINGBOARD, offered as A, B, C, and D.

With the notable exception of the Apple IIgs, the standard Apple II machines never had particularly good sound, all an Apple II programmer could do was to form sounds out of single clicks sent to the speaker at specific moments, which made the creation of complex sounds extremely difficult to program and made it mostly impossible to do any other processing during the creation of sounds.

With the arrival of the MOCKINGBOARD, this expansion card allowed programmers to send complex, high-quality sound via its specialized hardware, without need for constant CPU attention. The card produces sound through external speakers.

The MOCKINGBOARD was available in various models for either the slot-based Apple II, Apple II Plus, Apple IIe systems or in one special model for the Apple IIc.

Sound was generated through one or more AY-3-8910 or compatible sound chips, with one chip offering three square-wave synthesis channels. The boards could also be equipped with an optional speech chip (a Votrax SC-02 / SSI-263 compatible).

### 2005

The [ReActiveMicro MOCKINGBOARD](#) made available in 2005 (then called GSEReactive) was the first company to reproduce a clone of the board called “MOCKINGBOARD v1”.

Since then ReActiveMicro has released several versions of the MOCKINGBOARD in both ready to use and in assembled kit form.

#### Minimum System Requirements:

- Apple II, IIplus, IIe, IIgs
- 48K RAM
- 1 Disk Drive (or Equivalent)
- Monitor
- 2 External 8ohm Speakers

## INTRODUCTION TO THE MANUAL

In order that you may progress at your own pace, this manual has been separated into sections. The first section is Speech. Here, you will learn to develop different personalities for MOCKINGBOARD using interesting expressions and voices. If you would like to, delve deeper and learn to correct mispronounced words.

The second section is Sound. learn to develop, modify or completely change sound effects. Take a ping and change it to a plunk or start with an explosion and change it into a train. Learn about how MOCKINGBOARD produces sounds and how to create them from scratch.

The third section is Programming. Learn how to include sound and speech you created into your own programs. if you don't have a program in mind, you will after you explore the many possibilities for sound and speech enhancements.

## CONTENTS

1.	INSTALLATION .....	1
2.	SETUP.....	2
3.	SWEET MICRO SYSTEMS SOFTWARE.....	3
	Direct Downloads .....	3
4.	DIRECTOR & COMPOSER .....	4
5.	POWER OF SPEECH.....	5
	Add Inflection with the Director's Cues.....	5
	Add Texture to the Voice .....	10
	Soft to Loud Voice .....	10
	Control A (^A) Amplitude .....	10
	Low to High Voice .....	11
	Control I (^I) Inflection .....	11
	Slow to Fast Voice.....	12
	Control R (^R) Speech Rate .....	12
	Alter the Voice Quality .....	13
	Control F (^F) Filter Frequency .....	13
	Pulling it All Together.....	13
	Save the Words Created.....	14
	Control S (^S) .....	14
	The Rule Table .....	14
	A Word About Phonemes .....	15
	How to Make Changes to the Rule Table .....	16
	Let's Hear It.....	18
	TEST MODE.....	18
	Locate the Source of the Problem .....	19
	How to Read a Rule .....	20
	Create a New Rule .....	23
	Make Corrections .....	26
	Save the Changes.....	27
	Delete a Rule.....	28
	Other Useful Commands .....	29
	One Final Instruction.....	30
6.	SOUNDFX.....	31
	A Few Words About the Sound Utility .....	31
	What You Need to Know About Sound.....	32

	MOCKINGBOARD “Knobs” .....	32
	Getting Acquainted .....	34
	Noise Only Sound Effects.....	38
	Turn On Noise Only .....	39
	Setting Amplitude .....	40
	What Makes A Train Sound Not a Gunshot?.....	40
	How Fast is the Train Going? .....	41
	Change a Train into a Helicopter .....	42
	Change a Helicopter into a Gunshot.....	42
	Tone Only Sound Effects .....	42
	Start Fresh with a Clean Screen.....	43
	Turn On Tone Only .....	44
	Set the Amplitude.....	44
	Create Musical Notes.....	44
	Play a Note.....	44
	Play a Chord .....	45
	Play Two Chords.....	45
7.	MOCK PROGRAMMING .....	47
	The Sound Chips .....	47
	Setting Sound Chip Parameters.....	48
	Summary of Primary Routines and Table Access Routine .....	50
	Siren Sound Effect .....	52
	The Speech Chips.....	55
	Using Text to Speech and The Rule Table in Your Program .....	58
8.	APPENDIX A Phoneme Chart .....	60
9.	APPENDIX B Programmable Sound Generator Registers .....	62
10.	APPENDIX C Noise and Tone Register .....	63
11.	APPENDIX D Envelope Shape Register .....	65
12.	APPENDIX E Equal Tempered Chromatic Scale .....	66
13.	APPENDIX F Assembly Language Program Listings .....	69
	Primary Routines for Slot 4.....	69
	Table Access Routine for Slot 4.....	70
	Processor Loop Sound Effect .....	71
	Speech Composite Driver .....	72
14.	COMPATIBLE SOFTWARE.....	74
15.	MOCKINGBOARD REVISIONS .....	75

## 1. INSTALLATION

- 1) Turn off your computer and remove its cover, leave computer plugged into power.
- 2) Discharge any static electricity by touching the metal power supply outer casing.
- 3) Hold MOCKINGBOARD by its edges. Avoid touching the gold plated edge connector. The oily residue from your fingers may contaminate the connector and cause electrical interference.
- 4) Extend the audio cable fully.
- 5) Connect the female Mini-Molex plug end of the cable to the audio cable connector located on MOCKINGBOARD.
- 6) The MOCKINGBOARD generally is installed in Slot 4. The 2 Pin Audio Cable can be connected to your Apple II motherboard and to the 2 pin header on the lower left of the MOCKINGBOARD labeled "MB Speaker". When you reboot or power on the Apple II you should hear the "beep" over your speakers. If you do not, reverse the 2 pin cable and try again. The sound is only able to transfer with the cable connected one way. The demo files were specifically written for slot 4.
- 7) For the Apple IIgs be sure to set the system on "Normal" speed in the Control Panel or the board will not work correctly.
- 8) Connect the RCA phono jack-ends of the audio cable to the speakers. MOCKINGBOARD has two ½ watt amplifier chips on board to directly connect it to your speakers. You may use an external amplifier. If you do so, connect the RCA phono jack-end of the cable to the stereo amplifier auxiliary inputs.
- 9) Once the card is installed properly, proceed to the setup.



## 2. SETUP

- 1) Assuming you have created disks from the Diskette images: [mockingboard1.dsk](#) and [mockingboard2.dsk](#), place the first disk in the floppy drive and boot the disk. Select SOUND EFFECTS DEMONSTRATION from the menu. This demonstration is presented in a menu form. You may hear the sound effects in any order and as many times as you wish. The sounds alternate between speakers, that is, the first selection is played on one speaker, the next sound effect is on the other speaker.
- 2) The BOMB sound effect jumps from one speaker to the other. Some, like CLOCK, play continuously until you select another letter. You may also play two sounds at the same time on separate speakers. Sounds A and B will play at the same time, but A and C will not because they use the same speaker.
- 3) To return to the Main Menu, press the ESC key.
- 4) Select A on the Main Menu for speech and select A again for a message from Sweet Micro Systems. If your board has the gift of speech, your message will be audible.
- 5) If your MOCKINGBOARD has a speech chip, select Text to Speech to type and have MOCKINGBOARD speak. When MOCKINGBOARD is ready, a question mark will appear at the top of the screen. Type any word or phrase and then press the return key. When MOCKINGBOARD has finished speaking what you have typed, the question mark will reappear. Type some more words or phrases. You may type as many as 239 characters at one time. To return to the Main Menu, type QUIT.
- 6) This demonstration disk contains other examples of MOCKINGBOARD's capabilities. In addition, utilities have been included to allow you to further explore both sound and speech.
- 7) Have fun.



### 3. SWEET MICRO SYSTEMS SOFTWARE

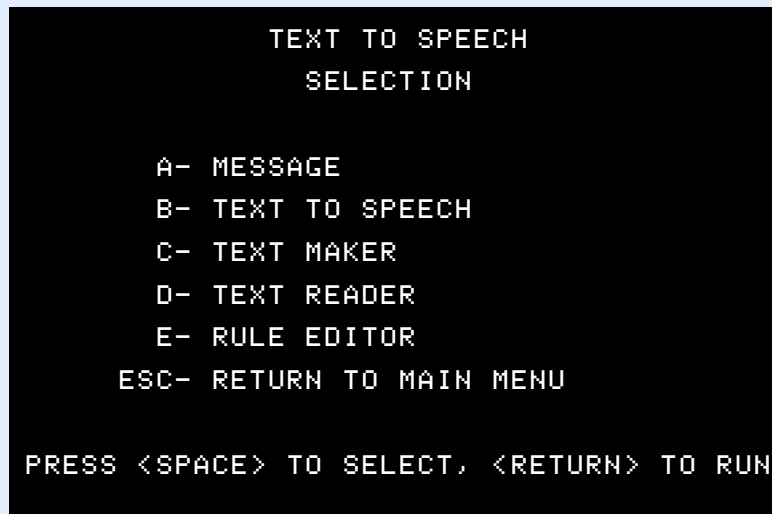
MOCKINGBOARD diskette images 1 and 2 include a wide variety of useful demonstrations and utilities. Also included are two speech related programs that allow you to enter unlimited words and have them spoken back.

The Text Reader program as seen in Display 1 will read back any text saved as a text file, not just those created using Text Maker. To use the Text Reader and Maker, please run Text Reader and enter the file name HELP. This file was created using Text Maker and will provide you with the necessary instructions to utilize both programs.

#### DIRECT DOWNLOADS



<a href="#">MOCKINGBOARD A Folder</a>
<a href="#">MOCKINGBOARD D Folder</a>
<a href="#">MOCKINGBOARD M Folder</a>
<a href="#">MOCKINGBOARD Sound &amp; Speech I Folder</a>
<a href="#">MOCKINGBOARD v1 Folder</a>
<a href="#">MOCKINGBOARD Disk 1</a>
<a href="#">MOCKINGBOARD Disk 2</a>



Display 1

## 4. DIRECTOR & COMPOSER

Think of MOCKINGBOARD as the phone of a friend you wish to call. In order to call a friend, you need the area code and phone number. During the conversation, information flows back and forth. Communication with MOCKINGBOARD is almost as simple as calling someone on the phone.

The concept of communicating to MOCKINGBOARD is often over looked. When you ran the demonstration, you made sounds flow from your speakers just by pushing a few keys. Trying to duplicate these sounds in your own programs takes a bit more effort. The programming is simple; the concepts behind it are not.

Telephones are connected to each other by lines through which our conversations are transmitted. Since we share lines with others, a direct line must be opened specifically for the two phones. This line is opened when you dial a phone number and someone picks up that phone. Communicating to MOCKINGBOARD works the same way.

The circuitry in your Apple allows you to transmit information directly to MOCKINGBOARD, but a direct line must first be opened. The slot number in which MOCKINGBOARD resides can be thought of as an area code. The "phone number" pinpoints a chip on MOCKINGBOARD.

MOCKINGBOARD has two chips which have their own "phone number" and may be "called" individually. These chips act as a switchboard routing all in-coming and outgoing "calls" to and from the Apple and MOCKINGBOARD's sound and speech chips.

You are the director/composer of MOCKINGBOARD sounds and speeches. This manual will step you through their creation and give you the necessary background information required along the way. First, we'll create sound/speech using the utilities provided on the demonstration disk. Then, we'll show you how to use the sounds created in a program.

## 5. POWER OF SPEECH

MOCKINGBOARD gives your Apple II computer the power of speech. Like a human, it will read text aloud pronouncing each word according to a series of rules. These rules are the basis for Sweet Micro Systems' method of converting text into a code MOCKINGBOARD can understand.

MOCKINGBOARD will allow you to introduce expression into the voice. Expression is important to the intelligibility and the meaning of the words spoken. The Sweet text to speech program automatically sets the speech parameters for general use and allows you to introduce stress and intonation to text by using special markers. You may change these parameters to create interesting voices.

There are many exceptions to standard pronunciation rules. Names are especially difficult and are frequently mispronounced. Remember how your teacher stumbled through her class list on the first day of school? How disappointed you must have been if your name was incorrectly announced to the class!

If MOCKINGBOARD has trouble with your name or names of family members and friends, you can easily correct it and we'll show you how. We know the name, Robert, is mispronounced. We will step through the corrections necessary, and in the process tell you about MOCKINGBOARD's features, capabilities and our method of converting text into speech.

A special section explains how to enhance your programs with speech you create using the Rule Editor or using the text to speech program right in a program of your own creation.

### ADD INFLECTION WITH THE DIRECTOR'S CUES

MOCKINGBOARD is all set to start talking. With a little assistance, MOCKINGBOARD will express itself with the use of inflection or pitch patterns, and show emotion. Limited use of inflection is automatically performed by the program. For example, it recognizes punctuation marks and responds accordingly. You will be able to employ inflection more creativity as you compose your sentences.

Boot your demonstration disk and select A for speech. Now select E for the Rule Editor from the Text to Speech Selection Menu. You will be asked to SELECT CHARACTER TABLE TO EDIT. Type A and the A rule table will appear. Type T for Test Mode at ENTER COMMAND.

**Now we are ready to proceed.**

```

=====
      TEXT TO SPEECH TEST MODE
=====
?  ✖

=====
      CURRENT PARAMETERS
=====
11 -AMPLITUDE          232-FILTER FREQUENCY
8  -INFLECTION          8  -SPEECH RATE
=====

```

Display 2

The cursor, next to the question mark as seen in Display 2 is ready for you to type in a word. After you type the word, press the return key. The word will be spoken at an average speed, in an average voice with minimal variation or emotional coloring. These speech characteristics have been preset to normally used values. If you would like MOCKINGBOARD to be more expressive, you may take advantage of its interpretive talents. MOCKINGBOARD's theatrical abilities are not to be underestimated.

Fine actors, regardless of their talent, require good directors. MOCKINGBOARD may be directed by inserting special markers into the text as it is typed in, these markers will tell MOCKINGBOARD when to show emotion, it already recognizes normal punctuation marks, such as commas, periods and question marks, and will respond with an appropriate pause, or raise or lower its voice. You may also place emphasis on a particular word or syllable, by inserting slash key stress markers (/) as cues to indicate when MOCKINGBOARD should play up a scene.

From the Test Mode, type the word HELLO and hit the return key. Think, like a good director, of the different ways that HELLO can be interpreted. When an actor speaks, he conveys emotion by changing the pitch, volume, and rate at which he speaks. Press return and listen.

How could you make this word more expressive? Try typing in the following examples. Each time you wish to clear an entry, type N for new entry. Should you wish MOCKINGBOARD to repeat itself, type R for repeat.

The comments to the right explain what effect the markers have on the word.

? HELLO	...would have no variation in stress
? /HE/LLO	...would stress HE
? HELL/O/	...would stress O
? HELLO?	...will cause a rise in pitch at the end
? HELLO.	...will cause a drop in pitch at the end

---

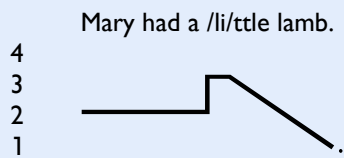
**Note:** You do not have to type the question mark; it appears automatically on the screen.

---

Other combinations of punctuation marks and stress marks are also possible. Stress markers generally work in pairs, but you may insert any number of them into a text. The number of stress markers and their position will determine how each word or syllable will be spoken.

**Be experimental!**

## INFLECTION DIAGRAM 1: DECLARATIVE SENTENCE



Type the following:

? MARY HAD A /LI/TTLE LAMB.

MOCKINGBOARD has just described Mary's pet. Diagram 1 shows the inflection pattern, or the rate of change of pitch, for a basic declarative sentence, which emphasizes the lamb's size.

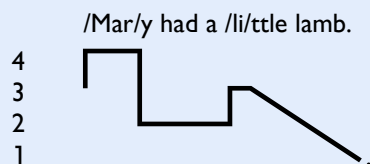
The English language has several levels of pitch. Our text to speech method approximates these levels by using four main pitch levels. These levels are designated by the digits which appear on the left side of the diagram.

"Mary had a little lamb" is spoken at pitch level two until the first stress marker is encountered. At the first stress marker, the pitch rises from level 2 to level 3. It will remain at level three until another marker is encountered.

At the second stress marker, the pitch will glide up or down depending on the final punctuation. A period at the end of a sentence, as in this example, indicates a drop in pitch.

If no final punctuation mark exists, then a period is assumed.

## INFLECTION DIAGRAM 2



If we want MOCKINGBOARD to show more feeling, we must give it additional direction.

Type the following:

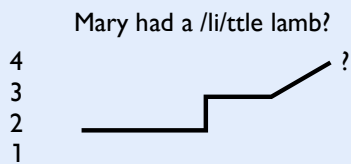
? /MAR/Y HAD A /LI/TTLE LAMB.

This diagram shows the change of pitch for a declarative sentence with more than two stress markers. In this example, the pitch starts at level 3 and rises to level 4 upon reaching the first marker.

The second marker signals a drop in pitch to level 2. Upon reaching the next pair of markers, the pitch level will again rise and then fall until the period is encountered.

MOCKINGBOARD's recitation of "Mary had a little lamb" deserves an ovation. The dual stress pattern was interpreted with greater emphasis on the first pair of markers than on the second. Such a stress pattern, in which the initial stress is more emphatic than stresses which follow, is typical of the English language.

### INFLECTION DIAGRAM 3: INTERROGATORY STATEMENT



Perhaps you would like MOCKINGBOARD to ask some questions about Mary? Let's change some of the cues and try some interrogatives.

Type the following:

**? MARY HAD A /LI/TTLE LAMB?**

If you compare this recitation with INFLECTION DIAGRAM 1, you will see that the performance differs only at the end where the different cue has caused a rise in pitch instead of a drop. The difference in pitch may appear to be insignificant, but we must remember that pitch assists us in interpreting a speaker's intent and helps us to recognize when he is stating or questioning. We are now doubting the lamb's small size.

The director's cues are actually much more sophisticated than they may appear. The stress markers not only cause MOCKINGBOARD to change its pitch, but also its volume, the number of words spoken per second and finally, the voice itself.

When a syllable is stressed, it generally becomes louder, the speech rate slows to make the syllable longer, and the voice quality changes slightly. You can achieve all of these theatrical effects simply by typing in normal punctuation and experimenting with the stress markers.



## ADD TEXTURE TO THE VOICE

As Director, you have only begun to utilize MOCKINGBOARD's many talents. With your assistance MOCKINGBOARD can change its voice.

MOCKINGBOARD's voice is described by four parameters as seen in Display 3: Amplitude, Inflection, Filter Frequency and Speech Rate. These parameters have been preset to values which will appear in the lower half of the Test Mode screen.

```
=====
                CURRENT PARAMETERS
=====
11 -AMPLITUDE      232-FILTER FREQUENCY
8  -INFLECTION     8  -SPEECH RATE
```

Display 3

To change any of these parameters, the commands shown in Table 1 will allow you to do so. The ^ mark represents the CONTROL key (or CTRL).

KEYBOARD	COMMAND	RANGE
CONTROL A (^A)	AMPLITUDE	0-11
CONTROL I (^I)	INFLECTION	0-25
CONTROL F (^F)	FILTER FREQUENCY	0-253
CONTROL R (^R)	SPEECH RATE	0-13

Table 1

---

**Note:** If you typed a word or phrase to be spoken and have not cleared it with an N for new entry, you will not be permitted to make any changes. The CURRENT PARAMETERS display is replaced by an ENTER COMMAND: prompt which will only accept R for repeat, N for new entry or Control S (^S) for save the word. Type N for new entry and you will be returned to an entry mode to make changes.

---

## SOFT TO LOUD VOICE

### CONTROL A (^A) AMPLITUDE

MOCKINGBOARD can speak in a variety of voices. It can speak in a barely audible whisper, or for stage purposes, in a deep sonorous voice. Volume or amplitude, may be adjusted by typing Control A (^A).

The program will respond with a prompt.

```
=====
ENTER NEW AMPLITUDE SETTING: *
=====
```

You may enter any setting from 0 to 11. The normal setting is set at 11. Try 4 and press return. The new setting will be reflected in the CURRENT PARAMETERS Table.

```
=====
CURRENT PARAMETERS
=====
4 -AMPLITUDE      232-FILTER FREQUENCY
8 -INFLECTION     8 -SPEECH RATE
```

Display 4

Now type in "Hello." If MOCKINGBOARD spoke too softly, type N for new entry, and A for amplitude. This time, try typing in 8, press return, and check for the new value in the Current Parameters Table as seen in Display 4. Direct MOCKINGBOARD to speak again. When you are satisfied that MOCKINGBOARD is speaking at a proper volume, you may turn your attention elsewhere.

## LOW TO HIGH VOICE

### CONTROL I (^I) INFLECTION

---

Different roles or personalities require different voices. A child speaks in a high pitched voice, an adult male in a low pitch. With your direction, MOCKINGBOARD can utilize its talents and do impersonations.

Suppose that MOCKINGBOARD was asked to play an evil villain in a theatrical production. Its normal voice won't do at all. In order to change pitch or inflection, type Control I (^I) in the Test Mode. A prompt will appear to assist you.

```
=====
ENTER NEW INFLECTION SET NUMBER: *
=====
```

You may enter any value from 0 to 25. When you change the inflection set, you are moving the four main pitch levels up or down on a musical scale. An evil character requires a very low voice, so let's type in 0, and press return. The new value will appear in the CURRENT PARAMETERS Table. Now type "/WEL/COME TO MY DOMAIN," press return, and meet your villain.

MOCKINGBOARD's talents are far too great to play only evil character types. Let's create another role. Type N for new entry, I for inflection and set the inflection set to 25. Press return. MOCKINGBOARD will now speak like a little child in a very high pitched voice. Type "M/OMM/Y?" and press return. MOCKINGBOARD's versatility will amaze you.

## SLOW TO FACE VOICE

### CONTROL R (^R) SPEECH RATE

---

Some roles will require that MOCKINGBOARD speak very quickly. The speech rate may be adjusted on a scale from 0 to 13, from excruciatingly slow to incredibly fast. Type Control-R (^R) for the prompt:

```
=====
ENTER NEW SPEECH RATE: 8
=====
```

Set the speech rate to 1 and press return. Also, type Control I (^I) for inflection and change it back to 8. MOCKINGBOARD's new line is, "I am s/o/tired," and it is spoken as though MOCKINGBOARD will be asleep before it reaches the word "tired."

(Don't forget to type the stress markers around the O.) On the other hand, type in a speech rate of 11, and press return.

Now type "Peter Piper picked a peck of pickled peppers," and press return.

**MOCKINGBOARD never stutters.**

## ALTER THE VOICE QUALITY

### CONTROL F (^F) FILTER FREQUENCY

---

The last parameter you may adjust is the Filter Frequency or voice quality. One of MOCKINGBOARD's greatest virtues is its ability to change its voice, if you type Control F (^F) in the Test Mode, the prompt will read:

```
=====
ENTER NEW FILTER FREQUENCY NUMBER:  ✖
=====
```

By typing in any number from 0 to 253, and pressing return, you may direct MOCKINGBOARD to speak in a different voice. Type in 242 and press return. Change the speech rate back to 8. Type "TAKE ME TO YOUR LEADER." MOCKINGBOARD could play a creature from outer space.

Let's try another. Type N and Control F (^F). Suppose we type 220 and press return. MOCKINGBOARD's voice acquires a previously undiscovered dignity.

If MOCKINGBOARD now says, "YOU ARE A/GREAT/ DIRECTOR," we can believe it.

### PULLING IT ALL TOGETHER

MOCKINGBOARD's abilities may be further explored by changing more than one parameter at a time. Try changing Filter Frequency and Inflection together. Any combination of the four parameters is possible, so you may create an unlimited number of voices.

Let's go back to the evil villain and make his voice more convincing. What the voice lacked earlier was the appropriate filter frequency.

Change the inflection to 0 and the filter frequency to 220, giving the speech a lower and deeper voice quality. Also slow the speech rate to 6. Now, type "/WELCOME/ TO MY DOMAIN. HA, HA, HA."

The child's whimper was high in pitch, but the voice quality was too strained. Change the voice quality to produce a softer, more innocent cry. Type 20 for inflection, 240 for filter frequency and 2 for speech rate. You may also lower the amplitude, if you wish. Type "/MOMMY? /I/ LOVE YOU."

## SAVE THE WORDS CREATED

### CONTROL S (^S)

---

As you develop words or phrases using the above methods, you may wish to save them. While the words and speech parameters are still on the screen, type Control S (^S) for save.

DO NOT TYPE N FOR NEW ENTRY BEFORE YOU TYPE CONTROL S. This will erase your words. Remember that after you enter a word, the only acceptable commands are N for new entry, R for repeat and S for save. When you type S you will be asked to enter a filename.

```
=====
ENTER FILENAME:  *
=====
```

You may enter any filename up to eight characters in length beginning with a letter A-Z. The following message will appear while the new file is written to your disk.

```
=====
PLEASE WAIT - SAVING COMPOSITE FILE
=====
```

The words you save may be used for current or future programs you may wish to enhance with speech. Please refer to the section on programming information for samples and an explanation of how you incorporate speech into your work.

We have only whetted your appetite. With all the features presented in the previous pages, you may create whatever creature or character your imagination dictates. MOCKINGBOARD's talents are constrained only by your imagination.

## THE RULE TABLE

Sweet Micro Systems' method of converting text to speech is rule based. Words are broken into sound patterns, which are represented by rules. MOCKINGBOARD matches these rules to characters in words or phrases. When a match is made, MOCKINGBOARD speaks.

The quality of rules developed in each character table will determine the accuracy of the resulting speech. Our language presents a formidable challenge in developing a comprehensive rule table. The Sweet table should be considered a base rule table, which may be personalized to suit your particular application.

Sweet Micro Systems has made an effort to free you from a predetermined vocabulary and pronunciation, by including a utility called the Rule Editor. The Rule Editor will allow you to alter the Sweet table. New rules may be added, existing rules may be edited or redefined, and nonessential rules may be deleted from the tables. Personalize the Sweet table and let MOCKINGBOARD tell you what you want to hear.

## A WORD ABOUT PHONEMES

---

MOCKINGBOARD produces speech using a building block method of combining basic sound units called phonemes. In order to teach MOCKINGBOARD to speak intelligibly, we must train our ears to hear individual phonemes in our own speech. MOCKINGBOARD can produce 64 speech sounds in all, more than enough to reproduce any speech you care to hear.

Phonemes may be divided into two distinct categories, consonants and vowels. A list of MOCKINGBOARD's phonemes, codes, and a key to their pronunciation are provided in [Appendix A](#). The chart is divided into two tables, one for vowels and the other for consonants. The phonemes are listed in the first column of each table. Each phoneme has four possible codes, which allow the user to select different durations for each sound. By referring to the examples and experimenting with phoneme length, anyone can produce highly intelligible speech.

Depending on where you live, your pronunciation of certain words may vary from MOCKINGBOARD's pronunciation. You will find that some words pronounced by MOCKINGBOARD will conflict with what you would normally expect to hear. Don't hesitate to change the pronunciation of any word you wish. MOCKINGBOARD has a great capacity to learn.

Boot disk 1 (mockingboard1.dsk) and select A for speech and then select Text to Speech, type your name after the question mark displayed and press return.

How did MOCKINGBOARD do? If MOCKINGBOARD pronounced your name correctly great! If not, let's correct the rule table so MOCKINGBOARD will always get it right.

Type QUIT to exit the Text to Speech mode and select the Rule Editor. The Rule Table has been designed to generate correct pronunciation for a majority of words. It operates using a text to speech method which allows the computer to analyze text, much in the way a person talks. Should the computer not be informed about a particular rule for pronunciation it will, like a human, make mistakes.

Errors will occur because our alphabet is not an accurate representation of our phonemic system. There is not a one-to-one relationship between an alphabet letter and a particular phoneme. If you think back to your grade school days, you will remember the difficulties first graders have with the rules for silent e, the e which is not pronounced but signals a change in the preceding vowel.

## HOW TO MAKE CHANGES TO THE RULE TABLE

---

### SELECT CONTROL Z (^Z)

When the Rule Editor is ready, you will see the following prompt at the top of the screen.

```
SELECT CHARACTER TABLE TO EDIT: ✖
```

The Rule Table consists of all alphabet letters, all digits and their upper case symbols, and all punctuation marks. In order to demonstrate how to correct the Rule Table, we have selected the name, Robert, which we know is mispronounced.

Type R for the R character table, The R table will appear on the screen. It should look like Display 5. The first two lines tell you where you are in the rule table and the present status.

The number of rules **1**, Address **2**, and bytes **3** will constantly change as you edit the table.

Ten rules will appear on the screen at a time. If the character table contains more than ten rules, press the space bar to advance to the next ten. When you reach the end of the table, press the space bar to return to the first ten rules.



## Screen Display of a Character Rule Table

```

0 RULE TABLE - R                1 NUMER OF RULES - 20
=====
2 ADDRESS - 35390                3 LENGTH - 203 BYTES
=====
4
1 1(R)! = 0E5C
2 1(RE(AC)!) = 1D01
3 1(READY) = 1D4A4A2501
4 1(READ) = 1D414125
5 1(REC) += 1D0130
6 1(REC) = 1D0A29
7 1(RE) ^# = 1D01
8 1(RE) 0 = 1D0A
9 1(RHO) M = 1D0E
10 1(RHO) = 1D1163
=====
5 ENTER COMMAND: *
=====

```

Display 5

## Key to The Rule Table

- 0** Indicates which character table you are viewing.
- 1** Indicates the total number of rules contained in this table.
- 2** Indicates the starting address in memory where this table can be found.
- 3** Indicates the total length (in bytes) of this table.
- 4** The first ten rules.
- 5** Type one of the editor commands listed in Table 2.

## List of Rule Editor Commands

KEYBOARD	FUNCTION	KEYBOARD	FUNCTION
U	Update Main Rule Table	CONTROL Z (^Z)	Select new character table
E	Edit an entry	CONTROL S (^S)	Save Rule Table to disk
I	Insert a new rule	CONTROL L (^L)	Load Rule Table
D	Delete an entry	CONTROL P (^P)	Print Character Table
T	Test node	CONTROL Q (^Q)	Quit or exit program
		CONTROL X (^X)	Help menu
SPACEBAR	Advance to next page of current Character Table		

Table 2

## LET'S HEAR IT

---

The Rule Editor has a test mode which allows you to evaluate MOCKINGBOARD's pronunciation of a word or phrase. You will be able to access this mode from any character table, and once in this mode, you may type any word or phrase.

## TEST MODE

---

Type T for the Test Node and a screen similar to that of Display 6 will appear. The Test Mode will allow you to enter 239 characters or about six and a half lines of characters at the question mark prompt. A beep will tell you that you have reached the limit. Type the letter U until you hear a beep. Press return and listen to the results. The sequence of two digit numbers at the lower half of the screen are the phoneme codes selected from the rule table by the text to speech conversion program. When you typed the return, the U's were converted to code using the rule(s) matching this character string. The Test Mode list of commands is listed in Table 3.

### Test Mode Screen Display

```

=====
      TEXT TO SPEECH TEST MODE
=====
?  ✖

=====
      CURRENT PARAMETERS
=====
11 -AMPLITUDE      232-FILTER FREQUENCY
8  -INFLECTION     8  -SPEECH RATE
=====

```

Display 6

### Test Mode List of Commands

KEYBOARD	FUNCTION	KEYBOARD	FUNCTION
R	Speak again	CONTROL A (^A)	Set amplitude
N	New entry	CONTROL I (^I)	Set inflection
CONTROL S (^S)	Save word or phrase	CONTROL F (^F)	Set Alter frequency
CONTROL Z (^Z)	Return to Editor	CONTROL R (^R)	Set speech rate level
		CONTROL X (^X)	Help menu
SPACEBAR	Advance to next phoneme page		

Table 3

## LOCATE THE SOURCE OF THE PROBLEM

Type N, to clear the input area for a new entry. Type Robert next to the question mark prompt and press return. It sounds close, but not quite right. The sequence of two-digit numbers at the bottom half of the screen represents the phoneme codes selected for Robert.

If you compare each of these phoneme codes with those of the Phoneme List in [Appendix A](#), you will find that this name is pronounced as /ROWBFRT/ and not as /RAHBERT/, which is correct.

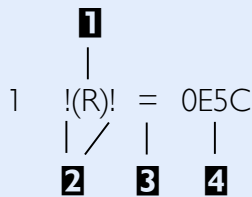
	1D	11	A3	64	SC	68	C0	
/	R	O	W	B	ER	T	PAUSE	/

In order to change the /OW/ sound to an /AH/ sound we must first determine which rule caused the error. Let us return to the rule table. Type N to clear for a new entry. Type Control-Z to return to the table from which you entered the Test Mode. Rather than go directly to the O rule table, we must first search the R rule table.

The rules in the R table always define how the letter R will be pronounced, but the next character(s) in sequence may also be included in the R rule. It is possible that a rule which exists for (RO) caused the error.

## HOW TO READ A RULE

Each rule in the table consists of three main parts, the rule definition on the left, the equals sign, and the phoneme codes on the right. The first rule of the R table states that **1** R, which is preceded and also followed by a non-alphabetic character **2**, is to be pronounced **3** as the composite sound of /AH-ER/, which is equal to the code **4** 0E5C.



- 1** Parentheses serve as boundary markers. They act to identify the particular character or characters which are to be matched. In this rule, only R will be pronounced.
- 2** The exclamation points indicate a non-alphabetic character which can be a space, punctuation mark, digit, or any other symbol except those which have been reserved as classification symbols (See Table 4 and Table 5).
- 3** The equal sign acts to assign the phoneme code to the contents of the parentheses.
- 4** If all the conditions on the left are met, then a match is achieved and the contents of the parentheses will be pronounced as indicated by the phoneme code(s) to the right. The codes are set aside in a buffer (a temporary memory location) until the entire word or phrase has been converted.

Other symbols used in rules are given in Table 4. The symbols help to generalize rules to encompass as many words with the same pronunciation pattern as possible. For example, a rule states that the letter A, preceded by any single consonant (^) and followed by the letter T, is to be pronounced as a short A. This rule may match the word BAT, CAT, FAT, HAT, NAT, PAT, RAT, SAT, etc. It will also match BATTLE, CATTLE, RATTLE, BATCH, CATCH, HATCH and so forth. This single rule will insure that the letter A, in all these words and many more like them, will be pronounced correctly.

How does the program know that B, C, F, etc. are consonants? The program is told. Each letter in the alphabet is classified as shown in Table 5. When Robert was typed, the program converted it to these symbols and set it aside for reference.

### Classification Symbols Used in Rule

Symbols for VOWELS		Symbols for CONSONANTS	
#	one or more vowels	^	one consonant
+	vowels E I Y	.	consonants BDGJLMNRVWZ
		:	zero or more consonants
Symbol for CHARACTER		Symbol for ALL OTHERS	
	use the character	!	Non-alphabetic

Table 4

### Classification Symbols: Used in Conversion

A	B	C	D	E	F	G	H	I	J	K	L	M
#	.	^	.	+	^	.	^	+	.	^	.	.
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
.	#	^	^	.	^	^	#	.	.	^	+	.

Table 5

Rule number 1 does not apply to Robert, because the O in Robert fails to match the exclamation point on the right of R.

```

/  R O B E R T  /
|  |  |  |  |
!  R  !

```

If we had typed in R alone, a match would have been achieved. The text to speech program automatically inserts a space on either side of a word or phrase to be converted, to mark where it begins and ends. Therefore, the exclamation point on the left matches the space which precedes the name, Robert. A match is not achieved on the right, because the letter O is a vowel, not a non-alphabetic character.

Compare the name, Robert, to the remaining rules in the R table. Each letter could be represented by its own character or a general symbol defining a vowel or a consonant. The letters in the name, Robert, may be represented by the symbols listed in Table 6.

R	O	B	E	R	T
.	#	.	+	.	^
^		^	#	^	
				:	

Table 6

Upon examination, we will see that a match will not occur until the last rule: (R)=1D.

The last rule states that R in any environment, excluding the rules preceding it, will be pronounced as the R in the word, “rat.” Rule number 16 only defines the pronunciation for the letter R, and not the sound of the letter O. Therefore, we must look to the next letter in sequence, the letter O, to locate the source of the mispronunciation.

### The O Rule Table

```

RULE TABLE - O          NUMER OF RULES - 84
=====
ADDRESS - 34039          LENGTH - 800 BYTES
=====
61 (O)^I#=1163
62 (O)^ICE=51A3
63 (O)^L#=11
64 (O)^R#=11
65 (O)^U=11
66 (O)^Y=11
67 (OUGHT)=1028
68 (OUGH)=505034
69 (OULDER)=116360255C
70 (OULD)=955565

=====
ENTER COMMAND: ※
=====

```

Display 7

We now proceed to the O table, type Control Z (^Z) to select a new character table and then O. If you page through the O table looking for a match, you should find a page of rules similar to Display 7.

A quick glance over the table indicates that all of these rules, with the exception of number 70, define a sequence of letters in which O must be followed by one consonant. To the left of O, no symbol or character exists. This means that the rule is not affected by what precedes O and this position is left unconstrained.

So far any rule from 61 to 69 could match Robert. Since it does not matter what precedes O and it is the only character within parentheses, we check for a consonant to the right and find B. To the right of B is the vowel E. Search the rules, starting with 61, for ^E, ^+ or ^#. Rules 61 and 62 can be eliminated since A follows. Rule 63 matches the E. Since the rule boundary ends here, a match is made.

```

/  R O B E R T  /
   | | |
   O ^ E

```

This rule states that whenever an O is followed by any single consonant and the letter E, the O will be spoken as the O in boat. If we try to change this particular rule so that Robert is pronounced correctly, we will find that this change affects other words, such as ROBE, ROPE, VOTE, and HOTEL. In order to avoid the possibility of such a side effect, let us create a rule just for Robert, since it appears to be an exception to this rule.

## CREATE A NEW RULE

---

In order to create a rule, we first have to decide where to place it. the placement of a rule is very important, not only within a character table, but also among the rule tables. Always place your rule in the table represented by the first character to be pronounced (within the parentheses). Since the purpose of creating the rule is to insure that the name Robert will be pronounced correctly, we will enclose all the letters within the parentheses. This rule will be placed in the R Table.

### INSERT

Type CONTROL Z (^Z) to select a new character table. Type R and the R rule table will appear on the screen.

First, we must determine where this new rule should be inserted. The program will search through the tables sequentially in its conversion process, so it is important that all exceptions be listed before the general case. Otherwise, the search may end prematurely with a rule for a more general case. We could not, for example, place Robert at the end of the table after the following.

```
20 (R)=10
```

If we tried to do so, our search would end with the above rule. This is a default rule which will match any word with an R since it does not specify what is to the left or right of R, the program would proceed to the next character search without ever reaching our Robert rule.



In the event that you are working with a table of many exceptions, it is wise to alphabetize the exceptions without violating the exception to general case order. In this manner it is easier to locate and examine a particular rule.

Since only the name Robert will match the rule we wish to create, it may be placed anywhere as long as it is before the last rule. For this example, let's place it in alphabetical order. Search through the table and find:

```
14 !(<RI>)U=1007
15 TH<ROUGH>=1016
```

The Robert rule could be placed between these two rules.

---

**Note:** This is an example. These rules may not appear in this manner or consecutively

---

Now that we know where we would like to place the rule, let's write it. Type I for Insert. You will be prompted with the instruction.

```
=====
ENTER RULE TO INSERT AT PROMPT BELOW...
=====
> ⌘
```

Type the first part of the rule as it appears below next to the > prompt. DO NOT PRESS RETURN! If you did press return, just press return again to display the Enter Command prompt, and begin once more by typing I for Insert.

```
=====
ENTER RULE TO INSERT AT PROMPT BELOW...
=====
> ⌘!(<ROBERT>!
```

If while entering the rule, you make a typographical error, you may backspace using the left arrow key and correct the error. However, if you type past the equal sign, you will not be permitted to back past it. If this happens, press the return. Press it again in response to the next prompt in order to cancel your entry. No rule will be inserted until you type in the location to insert. Now type the equal sign, DO NOT PRESS RETURN!

The exclamation points in this rule represent spaces. In this way we may exclude the possibility of altering the pronunciation of the same sequence of letters which may happen to be contained in a larger word. If, for example, we write a rule for the name, ROB, and leave both sides of the word unspecified, it would affect the pronunciation of words such as stROBe. To avoid this, we may define a space to the left and right, !(ROB)!, so that only these three letters would match this rule. Even ROBe would not match, since there are more letters to the right.

Refer to the phoneme page list and look for the phoneme code for an /AH/ sound to pronounce Robert correctly. A portion of that table has been reproduced below. Sometimes there may be more than one possibility. The list of phonemes contains two /AH/ sounds, specified by the phoneme codes beginning with 0E and 0F.

PHONEME LIST (PORTION)					
PHONEME	CODE				EXAMPLES
	1	2	3	4	
AE	0C	4C	8C	CC	dad
AE1	0D	4D	8D	CD	laugh
AH	0E	4E	8E	CE	top, about
AH1	0F	4F	8F	CF	father
AW	10	50	90	D0	saw, caught
AE	0C	4C	8C	CC	dad
AE1	0D	4D	8D	CD	laugh
AH	0E	4E	8E	CE	top, about
AH1	0F	4F	8F	CF	father

Table 7

Notice, that for each sound in the phoneme list Table 7, there are four possible phoneme codes. As the value is changed from that of column 1 to columns 2, 3, or 4, the duration of the sound is shortened by approximately twenty-five percent, you may select the length which sounds best to you. If you wish to lengthen a sound, place two phoneme codes for the same sound together.

Try the /AH1/ sound from the first column. Type the codes as indicated below, replacing only 11 and A3 with 0F for the 0 sound. The rule to be inserted should appear as follows:

```
=====
ENTER RULE TO INSERT AT PROMPT BELOW...
=====
[>]!(ROBERT)!=100F645C68C0
```

All phoneme codes are comprised of two digits. Leading zeros are necessary. Should you make an error, you will be allowed to backspace over the phoneme code? The backspace works a little differently with phoneme codes. A single backspace will move back and erase two digits rather than just one. This will prevent you from entering odd numbers of phoneme code digits. You will only be permitted numbers and the letters A-F on the right side of the equal sign.

Now press return, if you have not already done so, the program will ask you where you would like to insert the rule:

```
=====
ENTER BEFORE RULE NUMBER: ❷
=====
❶!(ROBERT)!=1D0F645C68C0
```

Insert the Robert rule before rule 15 TH(ROUGH)=1D16. This new rule will now be part of the table. The Editor will return to the first page of the table after inserting the rule. Press the space bar and find the new rule 15.

## MAKE CORRECTIONS

---

Now, let's hear it. Type T for the Test Mode. Type Robert after the question mark prompt and press return. How does it sound? It sounds much better, but let's try the other /AH/ sound, OE. Type N for New Entry and CONTROL Z (^Z) to return to the R table.

### EDIT

To make changes to a rule, type E for edit. You will be prompted with the following:

```
=====
ENTER NUMBER OF RULE TO EDIT: ❷
=====
```

Type in the number of the rule, 15. Press return, The Robert rule will now appear at the bottom of the screen above a prompt, so that you may refer to it during the edit. The entire rule MUST be reentered, not just the corrections. Partially typed rules will replace the original rule, in the manner typed.

The rule number is not necessary. As was the case for the Insert command, any typographical errors must be corrected before the equal sign is typed. You will not be permitted to backspace beyond the equal sign. If you type the equal sign, complete the rule, press return and type E to begin again. The rule should be completed so that you will not have to reconstruct the entire rule from your memory.

Typographical errors on the right hand side of the equal sign may also be corrected using the backspace. Remember that in order to preserve the two-digit code for a phoneme, a single backspace will move back two digits, not one, and that you will only be allowed to type number s and the letters A-F. Retype the rule with 0E, in place of 0F.

```
=====
ENTER BEFORE RULE NUMBER: 0
=====
0!(ROBERT)!=100E645C68C0
```

Press return and the edited rule will replace the old one. The display will show the first ten rules. Press the space bar and make sure the rule was edited properly, Test it once more. It should sound better and more intelligible. Once you are satisfied that this new rule functions correctly, type U to Update the table.

## SAVE THE CHANGES

---

### UPDATE (U)

When you select a character table, this one table is copied into a buffer area. A buffer area is like a temporary work space or scratch pad. You may make additions, deletions and changes to the rules while they are in this area. Once you are satisfied that the character table is correct, the Update command replaces the old table with the new table. Eventually, all the character tables will be saved permanently to disk.

The buffer area can only hold one-character table at a time. if you select another character table, the current table in the buffer will be written over by the new table. Any changes made will be lost unless an update was performed. Therefore, if you would like to see another character table, and you are not sure if you updated the current table, type U to update. No harm will be done if you did update earlier or made no changes.

### SAVE CONTROL S (^S)

Once your work is updated, type Control S (^S) to save the new table on your disk. The following prompt will appear at the bottom of the screen:

```
=====
ENTER TABLE NAME: 0
=====
```

You have an option to save the corrections in the rule table you are currently working with or save them under another name and create a new rule table. If you would like to create a new table, enter any file name up to eight characters in length, beginning with a letter from A-Z and press return. If you want to save the corrections in the current rule table, type Control N (^N). No file name is necessary. The standard rule table, provided on the demonstration disk, is called MKB:RULE. After entering the name or N, the Rule Editor will respond with:

```
=====
PLEASE WAIT - SAVING RULE TABLE FILES
=====
```

## DELETE A RULE

---

If you find that you have no use for a Robert rule, you may delete it. Any rule in any table may be deleted with the exception of the last rule. Each table must have at least one rule.

### DELETE (D)

Assuming that you are still in the R table, type D for Delete at the ENTER COMMAND prompt. The program will respond with:

```
=====
ENTER NUMBER OF RULE TO DELETE: *
=====
```

Type 15 and press return. The screen will display the following prompt along with the rule you selected. The rule will appear near the bottom of the screen.

```
=====
CONTINUE WITH DELETION? (Y/N) *
=====
```

Every attempt has been made to avoid mishaps, so you must confirm your intentions. If you respond Y, the deletion will proceed and all the rules following this rule will move up one position. The display will revert back to rules 1-10. Scroll through with the space bar to make sure the correct rule was deleted. Also check the last rule number to confirm the new rule count at the top of the screen display.

If you do not want to delete this rule, respond N, and the ENTER COMMAND prompt will reappear.

## OTHER USEFUL COMMANDS

---

### LOAD CONTROL L (^L)

After you become more familiar with the Rule Editor, you may discover more interesting applications for the text to speech capabilities. For example, you may be interested in foreign languages and might like MOCKINGBOARD to speak, maybe German? Or perhaps, you are writing a program which could use some speech. The standard rule table may be too bulky to be used with your program.

The solution is to create a new rule table for your application. You don't have to give this one up to get another. The demonstration disk contains a semi-blank rule table called MKB:EMPT. It contains the required one rule in each character table. If you do not wish to start from scratch, you may use the standard rule table (NKB:RULE), edit it and save it under another file name. This is done with the Save (S) command.

You may select a new rule table from any rule table. When you select the Rule Editor from the main menu, the standard rule table (MKB:RULE) will automatically be loaded. Select any character table and type Control L (^L) at the ENTER COMMAND prompt.

```
=====
ENTER TABLE NAME:  *
=====
```

Type the name of the rule table you wish to access. When a rule table is saved, three files are saved, the table itself, the total length of the table, and an index used to locate the character tables within the rule table. When a rule table is saved, .TABLE, .LENGTH, and .INDEX are appended to the file name automatically. The load command will automatically load the appropriate files, including the suffix. Therefore, when you load a table, you need only type its name.

```
=====
PLEASE WAIT - LOADING RULE TABLE FILE
=====
```

This prompt will appear, then the screen will clear and the initial select prompt will appear. You are all set to begin working with your new table.

```
SELECT CHARACTER TABLE TO EDIT:  *
```

### PRINT CONTROL P (^P)

The Rule Editor provides a simple to use print command to print out the individual character rule tables. Turn on your printer and type Control P (^P). The following message will appear on your screen.

```
=====
PRESS <SPACE> WHEN PRINTER IS READY: ✖
=====
```

The print out will look similar to the screen display, except that all the rules will be printed out in succession. When it is completed, the ENTER COMMAND prompt will reappear.

### HELP CONTROL X (^X)

Will display all the commands for the Rule Editor as seen in Display 8. It will also display the commands for the Test Mode when you are in that mode.

```

D: DELETE      E: EDIT      I: INSERT
^Q: QUIT       ^Z: SELECT    T: TEST
U: UPDATE      ^P: PRINT     ^S: SAVE

```

Display 8

### QUIT CONTROL Q (^Q)

When you are finished with the Rule Editor and wish to exit the program, type Z to select a new character rule table and type Control Q (^Q).

### ONE FINAL INSTRUCTION

---

It is important to note that your idea of the correct pronunciation may not be that of your neighbors. Some will prefer to say tomahto, others tomato. The Rule Editor allows you to change rules to suit your listening pleasure. It's yours!

Go ahead and make MOCKINGBOARD say your own name.



## 6. SOUNDFX

The MOCKINGBOARD sound system generates a remarkable array of sound effects and music. It is a natural addition to any program because it introduces real-life action and excitement to silent images and text. It fills time or sets moods with background music, and captivates the youngest of users with familiar and recognizable sounds.

In a very short time, MOCKINGBOARD will be making interesting sound effects under your control. As a method of introducing each of the sound controls used to generate effects, we will create two different types of sound effects using the SOUND UTILITY program included on the demonstration disk. This step by step explanation will be followed by examples showing how to incorporate sound effects into BASIC programs. The programs are well documented with comments called REM (remark) statements to help you understand the purpose of each line, just type in the code as written.

### A FEW WORDS ABOUT THE SOUND UTILITY

The Sound Utility program, provided on the demonstration disk, will allow you to create sound effects without programming. In fact, all the sound effects on the demonstration disk were first developed with the utility, then saved and incorporated into the programs later.

Please boot the demonstration disk and select Sound Utility. The monitor or TV screen will look similar to Display 9. SPEAKER: 1, at the top left hand corner, refers to the speaker which is the source of the sound. Below this line are the parameters which generate the sound.

At the bottom of the screen is a menu of commands. The cursor can be moved to different parameters using the arrow keys and/or Control-J (^J) for down and Control-K (^K) for up.

**Additional information about this utility will be presented as needed.**

### Sound Utility Screen Display

SPEAKER: 1									
=====									
REGISTER			MAX			CHANNEL			
NAME			VAL	ALL	A	B	C		
=====									
TONE PER FINE		255	0	0	0	0			
ONE PER COARSE		15	0	0	0	0			
NOISE PERIOD		31	0						
ENABLE		63	0						
AMPLITUDE		16	0	0	0	0			
(FIX=0-15/VAR=16)									
ENVL PER FINE		255	0						
ENVL PER COARSE		255	0						
ENVL SHAPE		15	0						
ONE PER COARSE		15	0	0	0	0			
=====									
=====									
P=PLAY	B=SIMULTANEOUS	L=LOAD	X=CLEAR						
R=RESET	C=SPEAKER	S=SAVE	Z=END						

Display 9

## WHAT YOU NEED TO KNOW ABOUT SOUND

Sound is a common phenomenon which we hear and feel every day, yet most of us have not given it much thought. What distinguishes one sound from another? How can a sound be duplicated? With MOCKINGBOARD, differentiating sounds is a natural process of developing them. Some sounds evolve into familiar, common sounds. Others become beautiful, exotic or mysterious. This process of developing MOCKINGBOARD sounds is comparable to adjusting your television set to get a clear picture. But instead of turning control knobs, you type in control adjustments to tune your sound, (See Table 8).

### MOCKINGBOARD “KNOBS”

Sound Quality	On/Off Switch	Volume	Pattern
Tone Period	Enable	Amplitude	Envelope Period
Noise Period			Envelope Shape

Table 8

#### TONE PERIOD/NOISE PERIOD:

Sound quality may be pure tone sounds, like musical notes, or noise, like rushing air. The Tone Period adjustment ranges from high to low pitch. The Noise Period adjustment also ranges from high to low, but not in terms of pitch. A high Noise Period sounds like the hissing of steam, while the low period sounds like the roar of rockets.

**ENABLE:**

Turns on or off the tone or noise generating capability. This is important, because MOCKINGBOARD is capable of producing up to six different sounds.

**AMPLITUDE:**

Controls the amplification or volume of the sound. There are two amplitude modes, fixed and variable. Fixed level amplitude provides 16 different levels of constant volume. Variable level amplitude passes the amplitude control to Envelope Period and Envelope Shape which generate amplitude patterns.

**ENVELOPE PERIOD/ENVELOPE SHAPE:**

Most sounds have a recognizable pattern which repeats. The pattern you hear is the change in volume. A sound may become loud, holds its level and then fade or soften. Envelope Period adjusts the length of one pattern by expanding or contracting it. With Envelope Shape you may select from 8 different shapes or patterns.

## GETTING ACQUAINTED

The Sound Utility program is capable of loading and saving sounds you develop. The sounds found on the demonstration disk were created using this program. To get acquainted with both the Sound Utility and the sound parameters, let's load a few existing sounds and play them. Each sound effect may be identified as either a pure tone, pure noise or a combination of the two.

Type L for load. The following will appear in the area just above the commands:

```
ENTER SOUND NAME OR <C>ATALOG => *
```

Let's listen to a pure tone sound, Type PING. The parameters for this sound effect are loaded into their respective fields or locations on the screen for evaluation and modification, the screen appears as seen in Display 10.

```
=====
SPEAKER: 1
=====
REGISTER          MAX      CHANNEL
NAME              VAL      ALL      A      B      C
=====
TONE PER FINE     255      0      20      0      0
ONE PER COARSE    15      0      0      0      0
NOISE PERIOD      31      0
ENABLE            63      62
AMPLITUDE         16      0      16      0      0
<FIX=0-15/VAR=16>
ENVL PER FINE     255      0
ENVL PER COARSE   255      0
ENVL SHAPE        15      4
ONE PER COARSE    15      0
=====

=====
P=PLAY      B=SIMULTANEOUS  L=LOUD  X=CLEAR
R=RESET     C=SPEAKER       S=SAVE  Z=END
=====
```

Display 10

Type P for Play and listen to the sound. As its name suggests, it is a "ping" sound. This one, quick, pure tone sound may be adjusted for a longer duration. Move the cursor down to ENVL PER COARSE, using the down arrow key or Control-J (^J) and type 8. Type P for play and listen to the change. Try a few more changes to this value, type 20, type 2, etc. Notice that the larger the value typed, the more drawn out the ping is.

Conversely, if a smaller number is entered, the ping becomes a plunk; not only is it shorter in duration, but one might suspect that it is a different sound altogether. The difference between the sounds is relatively minor. It is characterized by a change in the rate of decay or gradual decrease in volume.

The difference between the decays of ping and plunk is similar to the difference between rolling a ball down an incline and pushing the ball off the back, which is a cliff-like drop. The ball rolling down the incline takes longer to reach the ground than the ball dropping off the back.

Let's find out what fixed level amplitude sounds like. Change the ENVL PER value back to 4 before proceeding. Move the cursor up to AMPLITUDE. Using the right arrow key, move it to column A and type 10. Play the sound. Type R to stop the sound or type 0 to turn off the volume. This time the sound was piercing. The pitch was high and irritating. Move the cursor to Tone Period Fine tune to lower this pitch. Move the cursor to the A column and type 244.

This value is a C note in the fourth octave (middle C). A chart of TONE PER values for each note is provided in [Appendix E](#).

Our ping sound can also be altered by changing the amplitude pattern or the ENVL SHAPE. We have already adjusted the length of a decaying sound, but a sound which decays is only one of 8 possible patterns from which you may choose. The diagram as seen in Table 9 illustrates the different patterns we may select. Try each one to establish a relationship, in your mind, between the sound pattern and its picture.




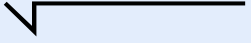

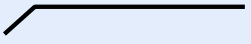

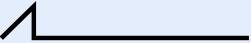
VALUE (DEC)	GRAPHIC REPRESENTATION
8	
9	
10	
11	
12	
13	
14	
15	

Table 9

Let's load a noise sound and play with it. Type L for Load, and type EXPLOSION. Type P to play. The ping sound is gone. We now hear a powerful blast. What parameters are required to make an explosion?

Move the cursor down to ENVL SHAPE. Notice the ENVL SHAPE is set to 0. Do not take 0 to mean that this parameter has been shut off! There is a sound pattern associated with a value of zero. This pattern is not included in the diagram above because it is the same pattern as for a value of 9.



SPEECH,  
STEREO  
MUSIC,  
SOUND  
EFFECTS

Change ENVL SHAPE to 14, and listen to your new sound. You are hearing the ocean, and all we had to do was change one parameter!

Move the cursor up to ENVL PER COARSE and type 10. Type P to play and listen to the difference. The ocean roar changed to a swish. Continue to reduce the value until you reach 1. This is a train sound. By changing only two parameters, we have created a wide range of special effects: explosions, ocean roars, swishes and trains.

Finally, let's listen to a sound which has both tone and noise. Type L for load and type ENGINE. The screen will appear as seen in Display 11.

SPEAKER: 1					
=====					
REGISTER	MAX		CHANNEL		
NAME	VAL	ALL	A	B	C
=====					
TONE PER FINE	255	0	0	0	0
ONE PER COARSE	15	0	5	5	5
NOISE PERIOD	31	10			
ENABLE	63	0			
AMPLITUDE	16	0	10	10	10
(FIX=0-15/VAR=16)					
ENVL PER FINE	255	0			
ENVL PER COARSE	255	0			
ENVL SHAPE	15	0			
ONE PER COARSE	15	0			
=====					
=====					
P=PLAY	B=SIMULTANEOUS	L=LOAD	X=CLEAR		
R=RESET	C=SPEAKER	S=SAVE	Z=END		

Display 11

Type P to play the engine sound. Tones are set at a coarse tune of 5 and noise is set at 10. The fixed amplitude is set to a moderate level of 10.

MOCKINGBOARD is capable of generating six different sounds at once. The three channels on each chip permit various combinations of tone and noise to be generated separately. The ENABLE parameter will allow you to designate whether a tone, noise, or a mixture of both is to be produced through each channel.

A chart in [Appendix C](#) contains values for the different possible combinations of sound. You may also find the proper ENABLE value in the Sound Utility by typing Control-E (^E). Enable is currently set at 0, let's find out what a zero setting means. Type Control E (^E), type 0 and type a return. A prompt will appear on the screen.

```
ENTER #0 TO 63 OR 'S' TO SCROLL => *
```

The chart just above this prompt tells you that all three channels are open for both tone and noise. If you type 63, all the channels are turned off. No sound will be generated. Press the ESC key and the cursor will return to where you typed Control-E (^E). Type 63 and P for play. MOCKINGBOARD is silent.

To hear just the tone component of the engine, type 56 for ENABLE. Verify that this number is for tone only in all three channels. Type Control E (^E) again and enter 56. Press ESC to return to ENABLE. Type P to play and listen to the result. The sound is a low pitched buzz.

Now type 7 for ENABLE and press P to play noise only in all three channels.

MOCKINGBOARD is producing a sound like television static. MOCKINGBOARD will allow you to mix these sounds just like an audio engineer. You make either tone or noise dominant by restricting the other to one channel only. for example, type 28 for ENABLE. The tone sound is now generated in channels A and B, while noise is restricted to channel C.

Experiment with other possible combinations, and listen to the difference. Further discussion and detail is given in the next two sections on developing noise only sound effects and tone only sound effects (musical notes).

## NOISE ONLY SOUND EFFECTS

You will have an opportunity to hear the difference between tone and noise, and the effects that can be created with them. We will begin with a noise effect – the sound of a train.

Using the Sound Utility, move the cursor to NOISE PERIOD. The Noise Period value ranges from 0 to 31. The value represents the amount of noise compressed within a period of time. The larger the value, the less noise compressed. The smaller the value, the more noise compressed. The result is similar to the sound of steam escaping from a kettle. The steam makes a high frequency hissing sound because the steam is trapped and is being compressed. If the lid is opened, the hiss immediately becomes lower in frequency because the steam is not being compressed.

The sound of a train is a soothing sound which is neither a high nor low frequency sound. Let's try a middle value. Type 16 next to NOISE PERIOD.



## TURN ON NOISE ONLY

Sounds produced by MOCKINGBOARD are routed through a passage, called a channel, to the speaker. Each sound chip on MOCKINGBOARD has three such channels for generating three separate sounds. MOCKINGBOARD orchestrates the three sounds and sends them out to a speaker. Since MOCKINGBOARD has two sound chips, it is capable of producing six different sounds, simultaneously, through two speakers.

Each channel may produce tone only, noise only or both. A single value entered for ENABLE will set each of the channels according to its representation.

A chart in [Appendix C](#), provides the values associated with all the possible combinations for three channels. We will work with three of the six channels to generate a train sound and designate each for noise only. The Sound Utility also provides this information if you request it. Move the cursor to ENABLE.

Type Control-E (^E) and the open box area near the bottom of the screen as seen in Display 12.

```
=====
ENABLE      NOISE      TONE
VALUE       C  B  A      C  B  A
ENTER #0 TO 63 OR 'S' TO SCROLL=> *
=====
P=PLAY      B=SIMULTANEOUS  L=LOAD  X=CLEAN
R=RESET     C=SPEAKER       S=SAVE  Z=END
=====
```

Display 12

Type S and what is shown in Display 13 will appear on the screen.

```
=====
ENABLE      NOISE      TONE
VALUE       C  B  A      C  B  A
0           ON  ON  ON    ON  ON  ON
PRESS SPACE TO CONT/ESC TO RETURN*
=====
```

Display 13

Press the space bar once and the A channel for TONE will change to OFF. Continue to press the space bar until all TONE channels are turned OFF and all NOISE channels are ON. The corresponding ENABLE VALUE is 7. Type ESC and the cursor will return to ENABLE at the top half of the screen or to where you typed Control E. Now type 7.

## SETTING AMPLITUDE

---

Move the cursor to AMPLITUDE. In order to hear the train sound, volume (or loudness) must be added. Amplitude levels can be set by either of two methods, a fixed level amplitude and variable level amplitude. With fixed level amplitude, a specific level of volume is selected and held constant until it is shut off or changed. This is similar to the volume control on a television set. You can adjust the volume to a comfortable listening level and then listen to that level for the rest of the evening. If you wish to change it, you must get up to adjust it. The fixed level is selected by setting AMPLITUDE to a level within a range of 0 to 15.

Variable level amplitude is more dynamic. That is, the loudness is not constant and may be altered at any time. Natural sounds are dynamic; each has recognizable characteristics. A bird's chirp cannot be mistaken for a dog's bark nor the patter of rain on a window. The chug of a train is also distinguishable from the whirl of the helicopter or the blast of a gunshot. It is a steady sound but not constant.

The train sound is characterized by a variable amplitude. The variable level is selected by setting AMPLITUDE to 16. Type 16 in the ALL column and all three channels will be set to 16. When the cursor is moved away from AMPLITUDE the value in the ALL column will revert to zero.

## WHAT MAKES A TRAIN SOUND NOT A GUNSHOT?

---

Change in amplitude distinguishes one sound from another. Amplitude may vary in any of three ways or modes, it can get louder (attack), it can hold at a particular level of loudness (sustain) and it can get softer (decay). Attack, sustain and decay are terms used to describe the amount of energy a sound gains or loses. For example, if you blow up a balloon and release it, the sound is very loud just as it is released. Almost immediately it begins to fade as the air escapes.

The amplitude pattern of the train sound (chug, chug) evenly increases and then decreases in volume; there is the same amount of attack as there is decay. If we were to draw a picture of this amplitude pattern, it would look like a zigzag. In contrast, a gun-shot sound does not repeat itself. It is one quick blast. It starts out very loud, and then fades (decays).

Conveniently, we do not have to create these patterns from scratch. MOCKINGBOARD can generate several common amplitude patterns.

These patterns, called Envelope Shapes or ENVL SHAPE, range from 0 to 15, but the actual number of different patterns is 8. These eight different patterns are illustrated in Table 10 and also in [Appendix D](#). Shapes 0-7 generate only one cycle of any sound. Although they are usable, they may be uninteresting. Concentrate on shapes 8-15.

## Envelope Shape Patterns


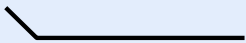






VALUE (DEC)	GRAPHIC REPRESENTATION
8	
9	
10	
11	
12	
13	
14	
15	

Table 10

Shape 14 describes the zigzag pattern of a train sound; it starts soft, gets loud, then soft and continues until it is changed or shut off. Set ENVL SHAPE to 14.

## HOW FAST IS THE TRAIN GOING?

The envelope establishes the basic amplitude pattern of a sound, but it is the duration of one cycle of the pattern which generates the effects. A train may be speeding along or slowing down. If the zigzag pattern of the train is compressed it will simulate a speeding train. A very loose zigzag will create the impression of a train moving slowly. The duration of a cycle is called an Envelope Period. Move the cursor to ENVL PER COARSE. ENVL PER COARSE stands for Envelope Period Coarse Tune.

The envelope period range is from 0 to 65,535; this is broken down into fine tune (0-255) and coarse tune (0-255). Fine and coarse tuning may be thought of in terms of minutes and seconds. It takes 60 seconds to equal one minute. Similarly, a value of 256 for fine tune equals one unit of coarse tune. The coarse tune will determine general duration of one cycle of a particular envelope and the fine tune will define the exact duration.

The envelope period of a speeding train is short, so set the ENVL PER COARSE to 1. Let's listen to it. Type P for play. Stop it by typing R for reset. Now, type P again, but this time do not type R. Experiment by typing in different values for ENVL PER COARSE and ENVL PER FINE to change the train's speed. Type P each time you wish to hear the result.

## CHANGE A TRAIN INTO A HELICOPTER

---

Move the cursor to ENVL SHAPE and change the pattern to 12. TYPE P to play. Listen, it sounds like a helicopter. A helicopter sound does not have a zigzag amplitude pattern. As the blade of the helicopter whips around, the sound builds from soft to loud and quickly drops back to soft. This sound pattern is an example of attack, there is no decay; the pattern resembles the outline of a saw-tooth. If you wish to slow the helicopter down, increase the ENVL PER COARSE

## CHANGE A HELICOPTER INTO A GUNSHOT

---

Type 0 for ENVL SHAPE and move the cursor to ENVL PER COARSE. Type P. It sounds like something dropped. Type 10 and type P. It sounds like a gunshot. A gunshot is a single burst of sound which does not repeat. It does not appear to use the ENVL SHAPE, only the ENVL PER COARSE. But in fact, the envelope shape, when set to 0, is an example of a decaying sound; it starts loud and fades in one cycle. Shape pattern 0 is a duplicate of shape 9. The ENVL SHAPE and ENVL PER COARSE and FINE are turned off only by a fixed AMPLITUDE setting of 0 to 15. Try it.

Most of the sounds heard on this disk can be loaded using the Sound Utility. Type L for load and type the name of the sound or C for catalog to see the list. Each sound parameter will be displayed and can be changed. If you develop a sound you would like to save, type S for save and type a descriptive name. Do not use the same name used to load the sound, save it under another name. If you save it under the same name, you will lose the original sound and replace it with your new sound.

## TONE ONLY SOUND EFFECTS

MOCKINGBOARD has a six channel sound capability and wide frequency range. Its full eight octave range makes it an ideal instrument for music composition. Two three note chords can be played at the same time, allowing you to compose songs with full accompaniment.

Our train sound was made up of noise only. It was generated by setting the Noise Period to some value from 0 to 31. In contrast, musical notes consist of pure tone sounds. They are generated by setting the Tone Period. The Tone Period value ranges from 0 to 255 for fine tune and from 0 to 15 for coarse tune.

Tone Period values represent the amount of compression or expansion of pure tone sound within a period of time. The smaller the Tone Period value the more compressed the sound; and therefore, the higher the pitch. Conversely, if the value is larger, the tone sound is expanded and the pitch is lower.

Fine and coarse tune are MOCKINGBOARD's tuning pegs. It is important to be able to obtain the frequency or the particular note desired, because sour notes are easily identified. In order to adjust the pitch on your MOCKINGBOARD, use the coarse tune to get the general frequency range and the fine tune to get the desired pitch. The relationship behind the fine and coarse tune is like the example given for Envelope Period. Coarse tune represents minutes of accuracy; fine tune allows us to determine the seconds. A fine tune of 256 is equal to 1 coarse tune.

A chart of the TONE PER values for each note is provided in [Appendix E](#). The fourth octave of this chart has been reproduced in Table 25 so you can easily select a few notes to play. Let's begin with the C-note. The decimal equivalent is 244.

#### Equal Tempered Chromatic Scale: Fourth Octave

TONE PERIOD				
NOTE	OCTV	NOTE FREQ	(DEC)	
			CRSE	FINE
C	4	261.624	0	244
C#	4	277.184	0	230
D	4	293.664	0	217
D#	4	311.128	0	205
E	4	329.624	0	193
F	4	349.232	0	183
F#	4	369.992	0	172
G	4	391.992	0	163
G#	4	415.304	0	153
A	4	440.000	0	145
A#	4	466.160	0	137
B	4	493.880	0	129

Table 11

#### START FRESH WITH A CLEAN SCREEN

The screen is still set for a noise sound and must be cleared before we begin making tone sounds. Type X for clear and respond Y, for yes, to the prompt appearing in the box at the lower half of the screen. All the values previously set have been returned to zero. Move the cursor up to TONE PER FINE by typing Control K (^K) or the up arrow key. Move the cursor over to the right once to channel A, by typing the right arrow key. Type 244.

## TURN ON TONE ONLY

---

Move the cursor to ENABLE and type Control-E (^E). The box in the lower half of the screen will prompt you to enter either a number or S for scroll. Type S to scroll and press the space bar until only channel A for tone is ON; all other channels should be OFF. The Enable value is 62. Type ESC and the cursor will return to ENABLE. Type 62 to enable channel A for tone only.

## SET THE AMPLITUDE

---

AMPLITUDE or volume of a sound may be set in two ways. The first way is to generate a fixed level amplitude, which produces a sound with a constant volume. The sound will remain at the same level until it is changed or shut off by an R for reset (stop). Fixed level settings range from 0 for no volume to 15 for maximum volume.

Move the cursor to AMPLITUDE and press the right arrow key to move the cursor to channel A. Set the AMPLITUDE to fixed mode by typing 15. Type P to play and listen. Type R for reset to stop the sound. The sound is a tone or a musical note but the constant flow of the sound is annoying. Let's try to set the AMPLITUDE in another fashion.

## CREATE MUSICAL NOTES

---

Variable level amplitude is controlled by a preset pattern of loud and soft levels. Variable level is selected by setting AMPLITUDE to 16. Under variable level amplitude, control is passed to the envelope shape and period controls. The changes in amplitude which distinguish one sound from another are called the envelope. This amplitude pattern describes the shape of the envelope (ENVL SHAPE). [Appendix D](#) shows the 8 different patterns available on MOCKINGBOARD.

The duration or envelope period (ENVL PER) of each pattern may also be controlled to create different effects. For example, a zigzag pattern, one which evenly glides up to loud and glides down to soft, may be compressed tightly so that the result is a tense and pulsating sound. The same zigzag pattern may be expanded. The sound is now calm and rolling.

## PLAY A NOTE

---

You may make the sound stop automatically, by setting AMPLITUDE to variable level and the ENVL SHAPE to a one cycle pattern. The pattern should start loud and glide down (decay) until no sound can be heard (pattern 0 or 9). The rate at which the note decays may be determined by the ENVL PER value. This capacity to control a note's rate of decay allows you to define the note as whole, half, quarter, eighth, etc.

Set AMPLITUDE for channel A to variable by typing 16. The ENVL SHAPE should be set to a decay pattern since musical notes naturally decay after an initial burst. A strike of a piano key or a pluck of a guitar string starts loud and fades as the vibration subsides. You may leave ENVL SHAPE at 0; patterns 9 and 0 have the same decay pattern. Move the cursor to ENVL PER COARSE and set the period to 20. This period value will play the musical note just long enough to allow it to decay rather than end abruptly. Type P for play.

## PLAY A CHORD

---

Now that we have learned to produce a note with proper decay, a chord can be built using the other two channels. Move the cursor back up to TONE PER FINE and over to the right to channel B. Type 193 for an E-note. Move the cursor to channel C and type 163 for a G-note. Move the cursor down to ENABLE and set it to 56, which enables all three channels for tone only. Move the cursor down to AMPLITUDE in the ALL column and type 16 for variable level. This will set the amplitude of all three channels to 16. Type P for play.

## PLAY TWO CHORDS

---

The Sound Utility also allows you to play six notes simultaneously by setting three notes on one screen and three on a second screen. The three notes you have just played appear on the first screen. Type C and a new screen will appear. Note that the parameters are cleared to zeros. Type C again to return to the first screen. The three notes are still there.

The screens are oriented to the speakers through which the sounds are played. The chord just played was created for SPEAKER: 1. This designation appears in the upper left hand corner. Type C and note that SPEAKER: 2, in the upper right hand corner, replaced SPEAKER: 1. The next three notes are also C, E, and G notes but on a lower octave, as seen in Display 14.

These note values include coarse tune values. Type the values as shown in Display 14.

SPEAKER: 2									
=====									
REGISTER		MAX		CHANNEL					
NAME		VAL	ALL	A	B	C			
=====									
TONE PER FINE		255	0	209	7	140			
ONE PER COARSE		15	0	3	3	2			
NOISE PERIOD		31	0						
ENABLE		63	56						
AMPLITUDE		16	0	16	16	16			
(FIX=0-15/VAR=16)									
ENVL PER FINE		255	0						
ENVL PER COARSE		255	20						
ENVL SHAPE		15	0						
ONE PER COARSE		15	0						
=====									
=====									
P=PLAY	B=SIMULTANEOUS	L=LOAD	X=CLEAR						
R=RESET	C=SPEAKER	S=SAVE	Z=END						

Display 14

Type P for play and listen to the deeper sound of this chord. Now, type B for both speakers and listen to six notes simultaneously.

---

**Note:** Both speaker designations appear at the top of the screen. If the sounds were continuous, R for reset would stop them. Since the notes decay automatically, a reset is not necessary.

---



## 7. MOCK PROGRAMMING

Whether you compose music, dialogue or design special effects, your masterpieces are unfinished until they are arranged in a production. What could be more original than to enhance one of your own programs with sounds of your own creation?

The sound chip and speech chip do not speak the same language. Therefore, instructions to any chip must go through a translator. This translator is called the 6522 Versatile Interface Adapter or just 6522. There are only four instructions for the sound chips and one for the speech chips which must be translated.

### THE SOUND CHIPS

The Sound Utility program demonstrated all the parameters needed to produce a sound. There are 16 in all: six Tone Period (Fine and Coarse), one Noise Period, one Enable, three Amplitude, two Envelope Period (Fine and Coarse), one Envelope Shape and two unused.

When all 16 parameter values are received by the sound chip via the 6522, sound is generated.

With the Sound Utility, the cursor is positioned over one of the 14 parameters and a value is typed. For Example, if we wish to set the Tone Period (Fine Tune) for channel A to 145, the cursor is moved to the first row, column A, and 145 is typed in. A similar location called a register address, is also designated on the sound chip for this parameter. Each parameter has its own register address numbered 0 to 15.

Table 27 shows each parameter and its associated register address. A chart of all registers and their descriptions is provided in [Appendix B](#).

## Sound Chip Registers

REGISTER ADDRESS	SOUND PARAMETERS	
0	Tone Period Fine Tune	for channel A
1	Tone Period Coarse Tune	for channel A
2	Tone Period Fine Tune	for channel B
3	Tone Period Coarse Tune	for channel B
4	Tone Period Fine Tune	for channel C
5	Tone Period Coarse Tune	for channel C
6	Noise Period	
7	Enable	
8	Amplitude	for channel A
9	Amplitude	for channel B
10	Amplitude	for channel C
11	Envelope Period Fine Tune	
12	Envelope Period Coarse Tune	
13	Envelope Shape	
14 & 15	Unused	

Table 12

## SETTING SOUND CHIP PARAMETERS

A sample program called TABLE Access Routine, provided on the demonstration disk, illustrates one method of setting sound parameters on the chip. Let's use our earlier example and set the Tone Period Fine Tune for channel A to 145.

Table 27 indicates that Tone Period Fine Tune for channel A is register address 0. The Table Access Routine is designed to send values to all sixteen registers sequentially, beginning with 0. It first sends MOCKINGBOARD the register address (0) of the data to be sent. Next, it sends MOCKINGBOARD the data (145). The Table Access Routine continues this process until all sixteen register addresses have been sent data. If a parameter does not need to be set, you should send a zero to that register address.

All information to MOCKINGBOARD flows on one of two direct lines. The first line sends only register addresses and data. But MOCKINGBOARD cannot distinguish addresses from data, because the addresses and data are all numerical values. The second line is used to send an instruction identifying what was sent on the other line.

Let's use our example to clarify the information flow to MOCKINGBOARD. When 0 is sent on the first line, the second line sends instructions to MOCKINGBOARD that the 0 is a register address. Next, 145 is sent on the first line and the instruction to MOCKINGBOARD that 145 is data, is sent on the second line. The two lines work almost simultaneously. In fact, information may be sent to MOCKINGBOARD continuously, because it flows in only one direction.

A separate program called PRIMary Routines, contains the four subroutines to handle the instructions to MOCKINGBOARD.

#### **PRIMary Routines subroutines:**

- INIT subroutine.** Initializes or primes MOCKINGBOARD to receive information. The initialization process is done once at the beginning of a program. MOCKINGBOARD must be initialized or sound parameter s sent to it will be ignored. Your program will continue, but no sound will be generated. MOCKINGBOARD does not have to be initialized again during the same program, unless the computer is shut off or the system is reset.
- LATCH subroutine.** Tells MOCKINGBOARD that a register address is being sent to it.
- WRITE subroutine.** Tells MOCKINGBOARD that data (a parameter value) is being sent.
- RESET subroutine.** Clears all sixteen registers to zeros. As a precautionary measure, the registers should be RESET at the start of a program to avoid generation of unexpected sounds.

These two programs, Table Access Routine and Primary Routines, work together to send the proper instructions and data to MOCKINGBOARD. The Table Access Routine orchestrates the task. While Primary Routines send instructions to MOCKINGBOARD, it is the Table Access Routine which selects the appropriate instruction to send.

To understand the principles behind these two programs, let's examine what happens in a mailroom. The parallel between a mailroom and MOCKINGBOARD will clarify how MOCKINGBOARD transfers data for sound generation.

MAIL	MOCKINGBOARD
The mail truck arrives each day at 9:00 am. The mail clerk is prepared to receive the mail at this time each day.	CALL the INIT routine at the beginning of your program. This routine prepares MOCKINGBOARD to receive information.
The mail is sorted into piles for each mail slot. Each mail slot belongs to a particular individual and is appropriately labelled.	Each sound is described by 16 sound parameters and those parameters correspond to a particular register on the sound chip. The registers, numbered from 0 to 15, correspond to the mail slots. Each sound parameter is assigned to a register just as a name on the mail associates it with a particular mail slot. The sound parameters are predetermined and stored in memory.
Before the new mail is placed into the mail slots, each slot is checked to make sure it is empty.	The Table Access Routine RESETs (or clears) all 16 registers as a precaution against unwanted sounds.
The mail is sorted in mail slot order and the slot door for the first pile is opened.	The Table Access Routine sends the register address to MOCKINGBOARD. It also tells the subroutine LATCH (Primary Routines) to send instructions to MOCKINGBOARD to “latch” onto that register so the data will know where to go.
The first pile is retrieved and placed in the opened mail slot.	The Table Access Routine retrieves the data for the first register from its place in memory and sends it to MOCKINGBOARD. It also tells the subroutine, WRITE (Primary Routines), to instruct MOCKINGBOARD to place that data into the latched register.
Each remaining pile of mail is processed in the same manner. The distribution of mail is completed when all the piles have been placed in their respective slots.	The next register is latched and the corresponding data is written to that register. When all 16 registers are filled, MOCKINGBOARD will generate a sound and send it to the speakers.

LATCH and WRITE are similar to the standard BASIC programming command, POKE. The address is identified first, then the data to be poked.

The data retrieval method used in the Table Access Routine is comparable to the standard programming commands, DATA and READ, where data is retrieved and read into the program for processing.

## SUMMARY OF PRIMARY ROUTINES AND TABLE ACCESS ROUTINE

The Primary Routines consist of four subroutines called INIT, LATCH, WRITE, and RESET. Each subroutine has a specific function in the transmission of instructions to MOCKINGBOARD. These subroutines utilize the instruction line to transmit information to MOCKINGBOARD. Each tells MOCKINGBOARD what to do with the information being transmitted on the address/data line.

The Table Access Routine coordinates the flow of information to MOCKINGBOARD by sending information on the address/data line and selecting the appropriate Primary Routines instruction for the address/data sent. As the name of the line indicates, only a register address or data is transmitted on this line.

The Table Access Routine also retrieves the data from a memory location where sound parameters are stored. The data can be POKEd into memory at a specific location by READing the data into your program with a Data statement. The data may be stored in any unused memory location, but the Table Access Routine must be told its location.

The data is POKEd in register address order. Conveniently, the order is sequential. Therefore, the data for the first register is POKEd into the first address of the memory location, then the data for the second register is POKEd into the next consecutive address and so forth until all sixteen have been POKEd. If more than one set of sound parameters are needed, the next set of parameters may be stored immediately following the first set.

When the second sound is to be played, the Table Access Routine must know where this set of sound parameters begin (start of first sound plus 16), The two sound chips on MOCKINGBOARD may be accessed separately.

Each chip uses the same method of instruction and address/data transmission.

At the beginning of this section, we mentioned that the only difference between the two sound chips was that each had its own "phone number." Therefore, both the Primary Routines and Table Access Routines have duplicate sets of code to access each separately. Included in the Primary Routines is another set of four subroutines called INIT2, LATCH2, WRITE2 and RESET2. The Table Access Routine also has a duplicate set of code labelled in a similar manner.

## SIREN SOUND EFFECT

Type this program right on your demonstration disk. (You may leave out the REM (remark) statement, as it serves only to explain the flow of the program.) It is the first sound of a siren. If you are using a fresh disk, you must copy the PRIMary Routines and the TABLE Access Routine onto your disk and run them either before you run the program or within the program itself. (Delete last line of TABLE.) Also, MOCKINGBOARD must be initialized at the beginning, with a CALL 36864 (INIT) and CALL 36908 (INIT2).

```

5 REM***DATA FOR SOUND; 145=FINE TUNE; 62=ENABLE; 15=AMPLITUDE
10 DATA 145,0,0,0,0,0,0,62,15,0,0,0,0,0,0
18 REM***STARTING ADDRESS FOR STORAGE OF SOUND DATA
20 A=33024
25 REM***READ AND STORE ALL VALUES FOR SOUND IN SEQUENTIAL ORDER STARTING AT LOCATION
   33024***WHEN X=0, IT IS ADDED TO "A" WHICH IS 33024 TO EQUAL 33024. PLACE THE FIRST
   DATA FROM LINE 10, 145, INTO LOCATION 33024
30 FOR X=0 TO 15: READ D
40 POKE A+X,D
50 NEXT
75 REM***USE TEMPORARY LOCATIONS 8 AND 9 TO POINT TO THE STARTING ADDRESS OF SOUND,
   33024; CALL TABLE ACCESS ROUTINE TO SEND DATA TO MOCKINGBOARD AND PLAY THE SOUND
80 POKE 8,0: POKE 9,129: CALL 32768
85 REM***USE DELAY LOOK FOR DURATION OF SOUND
90 FOR T=1 TO 342: NEXT T
145 REM***CALL RESET TO STOP THE SOUND AND CLEAR REGISTERS
150 CALL 36897
160 END

```

Save and run this program. Lines 5 through 50 place the set of sound parameters sequentially in memory, starting at 33024. Line 80 plays the sound for the period of time designated in line 90. Line 150 stops the sound with a CALL TO RESET.

**Now, let's add the second part of the siren sound.**

```

15 DATA 86,1,0,0,0,0,0,62,15,0,0,0,0,0,0
30 FOR X=0 TO 31: READ D
100 POKE 8,16: POKE 9,129: CALL 32768
120 FOR T=1 TO 342: NEXT T

```

Line 15 is the second sound data and line 30 is changed to READ 16 more data. Line 100 plays the second sound. The temporary locations, 8 and 9, are changed to point to the second set of sound data. The Table Access Routine is CALLED to send it to MOCKINGBOARD. Line 120 allows us to hear this sound for a given period of time.

Save and run this program, did you hear a siren?

This program demonstrates two manipulation techniques, duration (lines 90 and 120) and sequencing of two or more sounds (lines 80 through 120).

A delay loop follows each note because, otherwise, the notes would alternate too quickly for you to hear them. The only difference between the two notes is the tone period registers for channel A, refer to the chart on musical notes, [Appendix E](#). The first note is a middle or 4th octave A with a Tone Period value of 145. The second note falls between F# and G in the third octave; it has a Tone Period value of 342 or a coarse tune of 1 and fine tune of 86.

The changes necessary to create these two tones involve only two registers. Therefore, load the base parameters needed to create the notes and change the Tone Period registers as required to alter the note. In addition, lines 70 and 140 places the sound generation in a loop so that the siren will play five times. Here's how to do it.

```

10 DATA 0,0,0,0,0,0,0,62,15,0,0,0,0,0,0
20 A=33280
30 FOR X=0 TO 15: READ D
40 POKE A+X,D
50 NEXT
55 REM***SET THE POINTERS TO POINT TO LOCATION 33024
60 POKE 8,0: POKE 9,129
65 REM***REPEAT THE SIEN SEQUENCE FIVE TIMES
70 FOR Z = 1 TO 5
75 REM ***PLAY THE FIRST SOUND
80 POKE 33024,145: POKE 33025,0: CALL 32768
90 FOR T = 1 TO 342: NEXT T
95 REM***PLAY THE SECOND SOUND. CHANGE FIRST REGISTER TO 86 AND SECOND REGISTER TO 1
   SINCE TONE PERIOD =342, FINE TUNE = 342-INT (342/256)*256 AND COARSE TUNE
   TUNE=INT(342/256)
100 POKE 33024,86: POKE 33025,1: CALL 32768
120 FOR T=1 TO 342: NEXT T
135 REM***IF Z IS LESS THAN 5, PLAY THE SEQUENCE AGAIN
140 NEXT
145 REM***SHUT THE SOUND OFF
150 CALL 36897
160 END

```

Now, try this same program, but this time make use of the two sound chips and both speakers. The Primary Routines and Table Access Routine have secondary sets of routines to produce sounds out of the second speaker with the other sound chip. You will be able to play one tone through the left speaker and the other through the right.

The Table Access Routine uses a temporary memory location to store the beginning address of the sound to be played back. This stored address is referred to as a “pointer” because it points to where the sound data begins.

We have already used locations 8 and 9, in our siren example, to store the address for the first sound chip. The second sound chip uses Location 10 and 11. The address is divided into high and low bytes.

High byte is calculated by taking the beginning address and dividing it by 256. The resulting whole number is stored in location 9 or 11. Low byte is the remainder from the division and is stored in location 8 or 10. The siren sound effects program stores the data at 33024. When divided by 256, this address equals 129 with no remainder. The 129 is stored in 9 or 11 and 0 is stored in 8 or 10.

To produce the siren sound through both speakers, set the pointer locations, 10 and 11 to 33024. The pointer for the first sound chip, 8 and 9, also points to this same address. Both chips may utilize data in the same memory location because the data is not affected by use.

```
D <==> POINTER LOC 8,9 -> SOUND CHIP 1 -> SPEAKER 1
A
T
A <==> POINTER LOC 10,11 -> SOUND CHIP 2 -> SPEAKER 2
```

Change the following lines as indicated:

```
62 POKE 10,0: POKE 11,129
100 POKE 33024,86: POKE 33025,1
110 CALL 36897: CALL 32796
130 CALL 36941
```

Line 110, the CALL to 36897, is the RESET subroutine (Primary Routine) for the first sound chip. This CALL will shut off the first sound chip. If we do not shut off one of the chips, both would produce sound at the same time, and we would create a two-note chord. This CALL was not necessary in the earlier programs because both notes were generated by one sound chip. The second sound wrote over the register values of the first sound.

Line 100, the CALL 32768, is dropped and picked up in line 110 as CALL 32796, which is the Table Access Routine for the second sound chip. It plays the second note through the other speaker. Line 130 turns off the second sound with a CALL to RESET2.

---

**Note:** The pointers (8,9 and 10,11) do not have to be POKEd each time with the address of the sound data. The sound data has not moved from its location; therefore, its address remains the same.

---



```

10 DATA 0,0,0,0,0,0,0,62,15,0,0,0,0,0,0
20 A=33024
30 FOR X=0 TO 15: READ D
40 POKE A+X,D
50 NEXT
55 REM**SET BOTH SOUND CHIP POINTERS TO POINT TO THE ADDRESS OF THE SOUND DATA, 33024.
60 POKE 8,0: POKE 9,129
62 POKE 10,0: POKE 11, 129
70 FOR Z= 1 TO 5
75 REM**CHANGE THE SOUND DATA IN MEMORY TO THE FIRST SOUND AND PLAY IT THROUGH THE
  FIRST SPEAKER WITH A CALL TO THE TABLE ACCESS ROUTINE. (NOTE THAT 32768 IS THE
  ROUTINE FOR THE FIRST CHIP.)
80 POKE 33024,145: POKE 33025,0: CALL 32768
90 FOR T = 1 TO 342: NEXT T
95 REM**CHANGE THE SOUND DATA IN MEMORY TO THE SECOND SOUND
100 POKE 33024,86: POKE 33025,1
105 REM**TURN OFF THE FIRST SOUND CHIP WITH A CALL RESET AND PLAY THE SECOND SOUND
  THROUGH THE SECOND SPEAKER WITH A CALL TO THE TABLE ACCESS ROUTINE.
  (NOTE 32796 IS THE ROUTINE FOR THE SECOND CHIP.)
110 CALL 36897: CALL 32796
120 FOR T=1 TO 342: NEXT T
125 REM**CALL RESET2 TO SHUT OFF THE SECOND SOUND CHIP 130 CALL 36941
135 REM**IF Z IS LESS THAN 5, PLAY THE SEQUENCE AGAIN
140 NEXT
160 END

```

In this last example, one tone of the siren is played through one speaker and the other tone through the second speaker. Other variations are also possible.

## THE SPEECH CHIPS

The Text to Speech Rule Editor is used to develop words or phrases for use in your programs. Enter the Test Node from any character rule table and type in the word or phrase you wish to use. Adjust the pronunciation, using the parameter controls and stress markers. Once you are satisfied with the quality of speech, save the word or phrase by typing Control S (^S) from the Test Mode. Name the file using a maximum of eight characters, beginning with a letter (A-Z).

The word or phrase saved is a composite file of all the speech parameters. However, the composite file contains only data. This data is similar to the 16 sound parameters needed to produce a sound effect. While the sound parameters are finite, the speech parameters consist of four parameters for each phoneme code generated to produce the speech. If a word consists of 20 phoneme codes, then the composite file of that word contains 100 parameters (including the phoneme codes).

From the Rule Editor, a word or phrase is spoken using the Text to Speech Algorithm. Outside of the Rule Editor, another type of program must be employed to generate speech. The Text to Speech Algorithm is no longer necessary, because the conversion from text to phoneme codes has already been done and saved. A program called the Composite Driver, included on the demonstration disk, acts as a messenger and transmits speech codes to the speech chip from the composite data file.

The Composite Driver is similar in concept to the Table Access Routine. The Table Access Routine retrieves data and sends it to the sound chip. The Table Access Routine knows where the data is located by using a pointer. The Composite Driver also has a pointer which tells it where the speech data is located.

These similarities standardize the programming method employed in generating both sound and speech. Let's enhance a short program with speech. Load the following files in memory prior to running the sample program or at the beginning of the sample program:

### BLOAD COMPOSITE DRIVER

```
BLOAD Name, A Address
```

Name = composite data file name

Address = address in memory

The composite data file may be stored in any unused memory space.

The pointer location for the beginning address of the speech data is stored in location 249 and 250. The high byte of the address is stored in 250 and the low byte is stored in 249. If you are unfamiliar with high and low bytes, the high byte is obtained by dividing the address by 256 and the low byte is the remainder of this division.

If the data is stored at 33024, then the high byte is  $33024/256$  or 129. The low byte is zero since there is no remainder.

The following phrase was created using the Rule Editor and saved on the demonstration disk as PHRASE 1:

### WITH MOCKINGBOARD YOU'LL NEVER BE SPEECHLESS

Type in the program below on your copy of the demonstration disk. If you are using a fresh disk, copy the COMPOSITE DRIVER and PHRASE 1 files onto your disk.

```
10 HOME
20 D$=CHR$(4)
25 REM***LOAD COMPOSITE DRIVER
30 PRINT D$"BLOAD COMPOSITE DRIVER"
35 REM***LOAD THE DATA FILE PHRASE 1 AT LOCATION 35072
40 PRINT D$"BLOAD PHRASE 1, A 35072"
45 REM***TELL COMPOSITE DRIVER WHERE THE DATA FILE PHRASE 1 RESIDES. CONVERT 35072: HIGH
   BYTE-INT (35072/256) OR 137, PUT IT IN LOCATION 250; LOW BYTE-INT (35072/256) - 137
   OR 0, PUT IT IN LOCATION 249.
50 POKE 249,0: POKE 250, 137
55 REM***TELL COMPOSITE DRIVER TO BEGIN SPEAKING. COMPOSITE DRIVER IS LOCATED AT 27904.
60 CALL 27904
70 END
```

This program will speak the entire phrase and then end at line 70. In most cases, the program would not speak and then end, it would continue on with other tasks. If this program did not end at line 70, the speech could be interrupted prematurely by an input from the keyboard or program code. In order to protect against such an interruption, the program should check to determine if the speech chip is finished speaking.

When MOCKINGBOARD speaks, a busy flag is set. This flag is located at location 255. When MOCKINGBOARD is finished speaking, the flag is cleared, your program can monitor this flag. A standard BASIC programming command called PEEK may be used to look at the contents of this location.

Type the following lines into the above program. Delete line 70. Save the new version of the program and run it.

```
80 VTAB 6: HTAB 1:PRINT "WOULD YOU LIKE TO HEAR IT AGAIN? ";:GET A$
90 IF A$="Y" THEN GOTO 60
100 IF A$="N" THEN GOTO 120
110 GOTO 80
120 END
```

The question, "WOULD YOU LIKE TO HEAR IT AGAIN?" appears on the screen almost at the same time as the speech begins. If you respond to this question before the speech ends, you will interrupt it. Try it. Press the Y key several times in quick succession. The phrase is not allowed to finish until you stop pressing the key. To prevent this, insert the following lines in this program. Save and run it. You will no longer be permitted to interrupt the speech.

```
65 REM***CHECK TO SEE IF FINISHED SPEAKING. IF NOT, KEEP CHECKING UNTIL IT IS.
70 IF PEEK (255)> 0 THEN 70
```

To incorporate more than one phrase in your program, load each file at the beginning of the program. Load each one at a different location, so you will be able to call them at will. Change the pointers, 249 and 250, to point to the phrase you wish spoken just before CALLing the composite driver. Let's try it with PHRASE2, which says: I LOVE TO TALK. This time, let's print the phrases on the screen.

```

10 HOME
20 D$ =- CHR$(4)
30 PRINT D$"BLOAD COMPOSITE DRIVER"
40 PRINT D$"BLOAD PHRASE1, A 35072"
42 REM***LOAD THE SECOND PHRASE AT 36864
45 PRINT D$"BLOAD PHRASE2, A 36864"
47 PRINT "WITH MOCKINGBOARD YOU'LL NEVER BE SPEECHLESS"
50 POKE 249,0: POKE 250,137
60 CALL 27904
70 IF PEEK (255)>0 THEN 70
80 VTAB 6: HTAB 1:PRINT "WOULD YOU LIKE TO HEAR IT AGAIN?";:GET A$
90 IF A$="Y" THEN GOTO 60
100 IF A$="N" THEN GOTO 120
110 GOTO 80
120 VTAB 10: HTAB 1:PRINT "WOULD YOU LIKE TO HEAR ANOTHER PHRASE?";:GET B$
130 IF B$="Y" THEN GOTO 152
140 IF B$="N" THEN GOTO 170
150 GOTO 120
152 VTAB 10: HTAB 1:PRINT "I LOVE TO TALK!"
155 REM***CHANGE THE POINTERS TO POINT TO PHRASE2 AT 36864: HIGH BYTE = INT (36864/256)
    OR 144, PUT IN 250; LOW BYTE = 36864/256-144 OR 0, PUT IN 249. TELL COMPOSITE DRIVER
    TO SPEAK THIS PHRASE.
160 POKE 249,0: POKE 250, 144: CALL 27904
170 END

```

Save and run this program. If you would like to try out a different phrase, change line 40 and/or 45 to load your file. Don't forget to change the phrase printed on the screen in lines 47 or 152. Why not make all the prompts in this program speak?

## USING TEXT TO SPEECH AND THE RULE TABLE IN YOUR PROGRAM

The above programming method is a very convenient way to generate speech, provided you know what vocabulary will be required in your program. This is not always possible or desirable. You may wish to have a person using your program type in his own responses. These words could also be spoken by MOCKINGBOARD. However, unless the response is limited to a predefined vocabulary, words not previously coded will be left unsaid.

Another method of generating speech will allow MOCKINGBOARD to speak an unlimited vocabulary. This program incorporates the Text to Speech program included on the demonstration disk. It uses a table of rules to convert text into speech. The text may consist of characters typed from the keyboard or characters assigned to a string variable. It may also be text saved in a text file.

If your program is very large, this method may not be economically implemented, due to the size of the rule table. However, if you can anticipate the vocabulary that may be used in your program, including responses from the user, an empty rule table may be used to build a custom list of words. The empty rule table will allow you to enter only rules which may pertain to your program. If you prefer, you may also trim the current rule table to a size more suitable to your program and save the revised version under another name.

Any rule table may be included in your program along with Text to Speech. A sample program using this method is given below. Regardless of whether you are converting input from the keyboard or assigning it to a string variable within your program, you must assign the input to the variable MB\$. The program, which Text to Speech uses to retrieve the text data, looks for this variable. This program is called MB\$ GETTEXT.

```

10 HOME
20 D$=CHR$(4)
30 PRINT D$"BLOAD TEXT TO SPEECH"
40 PRINT D$"BLOAD MB$ GETTEXT"
45 REM***LOAD THE RULE TABLE
50 PRINT D$"BLOAD MKB: RULE. TABLE"
60 PRINT D$"BLOAD NKB: RULE. LENGTH"
70 PRINT D$"BLOAD MKB: RULE. INDEX"
75 REM***ASSIGN THE PHRASE TO BE SPOKEN TO MB$
80 MB$ = "WITH MOCKINGBOARD YOU'LL NEVER BE SPEECHLESS"
85 REM***TELL TEXT TO SPEECH TO BEGIN SPEAKING THE PHRASE
90 CALL 26123
95 REM***CHECK TO SEE IF FINISHED SPEAKING. IF NOT, KEEP CHECKING UNTIL IT IS.
100 IF PEEK (255)>0 THEN 100
110 END

```

MKB:RULE is the standard rule table designed by Sweet Micro Systems. You may replace this with your rule table file name. The TABLE, LENGTH and INDEX must be appended to your rule table file name. These files monitor the expansion and reduction of the rule table as well as where all the characters reside in memory. They are always updated and saved when you save a rule table.

If you wish to speak a response from the user change line 80 to:

```
80 INPUT "ENTER TEXT: "; MB$
```

The INPUT statement may be any question or prompt.

## 8. APPENDIX A PHONEME CHART

### LIST OF CONSONANT PHONEMES

PHONEME	CODE				EXAMPLES
	1	2	3	4	
B	24	64	A4	E4	bat, tab
D	25	65	A5	E5	dub, bud
F	34	74	B4	EA	fat, ruff, photo, laugh
HV	2A	6A	AA	EA	eh?
HVC	2B	6B	AB	EB	d(h)ouble
HF	2C	6C	AC	EC	hat, home
HFC	2D	6D	AD	ED	P(h)ad, fluff(h), black(h)
HN	2E	6E	AE	EE	hnh-hnh
J	31	71	81	F1	job, rage
K	29	69	A9	E9	kit, tick
KV	26	66	A6	E6	big, gag
L	20	60	A0	E0	lab, ball
L1	21	61	A1	E1	plan, club, slam
LF	22	62	A2	E2	bottl <u>e</u> , chann <u>e</u> /
M	37	77	B7	F7	mad, dam
N	38	78	B8	F8	not, ton
NG	39	79	B9	F9	ring, rang
P	27	67	A7	E7	pat, tap
R	1D	5D	9D	DD	rat
S	30	70	B0	F0	sat, lass
SCH	32	72	B2	F2	shop, push
T	28	68	A8	E8	tap, pat
THV	35	75	B5	F5	bathe, the
TH	36	76	B6	F6	bath, theory
V	33	73	B3	F3	vow, pave
W	23	63	A3	E3	why, quake
Y1	04	44	84	C4	you
Z	2F	6F	AF	EF	zap, maze
	00	40	80	C0	[pause]

The 9 columns of code for each phoneme allow you to alter the length of any sound, and choose the version which provides the most intelligibility and natural quality. Each successive column represents a phoneme which is approximately 25% shorter than its predecessor.

For most purposes, column 1 will serve as a standard value.

PHONEME	CODE				EXAMPLES	
	1	2	3	4		
A	08	48	88	C8	day	
A1	09	49	89	C9	care	
AE	0C	4C	8C	CC	dad	
AE1	0D	4D	8D	CD	laugh	
AH	0E	4E	8E	CE	top, ab <u>o</u> ut	
AH1	0F	4F	85	CF	fa <u>t</u> her	
AW	10	50	90	D0	sa <u>w</u> , caugh <u>t</u>	
E	01	41	81	C1	be <u>e</u> t, b <u>e</u>	
E1	02	42	82	C2	ad <u>ve</u> nt	
EH	0A	4A	8A	CA	l <u>e</u> g, sa <u>i</u> d	
EH1	0B	4B	8B	CB	sil <u>e</u> nt	
ER	1C	5C	9C	DC	th <u>i</u> rd, u <u>r</u> n, h <u>e</u> ard	
I	07	47	87	C7	s <u>i</u> t, b <u>i</u> d	
O	11	51	91	D1	bo <u>a</u> t	
OO	13	53	93	D3	p <u>u</u> t, p <u>u</u> ll, lo <u>o</u> k	
OU	12	52	92	D2	o <u>o</u> rb	
U	16	56	96	D6	bo <u>o</u> t, y <u>o</u>	
U1	17	57	97	D7	po <u>o</u> r	
UH	18	58	98	D8	c <u>u</u> p	
UH1	19	59	99	D9	c <u>i</u> rc <u>u</u> s	
UH2	1A	5A	9A	DA	na <u>t</u> ion	
UH3	1B	5B	9B	DB	na <u>t</u> ion	
FOREIGN SOUNDS						
AY	05	45	85	C5	fran <u>ç</u> ais	French
A	3A	7A	BA	FA	ê <u>t</u> re	French or
						umlauted A
						in German
E2	3E	7E	BE	FE	sch <u>ö</u> n	German
IE	06	46	86	C6	il	French
IU	14	54	94	D4	pe <u>u</u> t	French
IU1	15	55	95	D5	Go <u>ö</u> the	German
OH	3B	7B	BB	FB	men <u>ü</u> , t <u>u</u>	French
U	3C	7C	BC	FC	f <u>ü</u> hlen	German
UH	3D	7D	8D	FD	men <u>u</u> , t <u>u</u>	French
Y	03	43	83	C3	y	French
LB	3F	7F	BF	FF	il	French
R1	1E	5E	9E	DE	r <u>e</u> ponse	French
R2	1F	5F	9F	DF	richt <u>i</u> g	German

## 9. APPENDIX B PROGRAMMABLE SOUND GENERATOR REGISTERS

REGISTER		DESCRIPTION	DEC	HEX
R0	CHANNEL A TONE PERIOD	FINE TUNE	0-255	00-FF
R1		COARSE TUNE	0-15	00-0F
R2	CHANNEL B TONE PERIOD	FINE TUNE	1-255	00-FF
R3		COARSE TUNE	0-15	00-0F
R4	CHANNEL C TONE PERIOD	FINE TUNE	0-255	00-FF
R5		COARSE TUNE	0-15	00-0F
R6	NOISE PERIOD	ALL CHANNELS	0-31	00-1F
R7	ENABLE	NOISE/TONE	0-63	00-3F
R8	CHANNEL A AMPLITUDE	AMPLITUDE LEVEL MODE SELECT  FIXED = 0-15 VARIABLE = 16	0-16	00-10
R9	CHANNEL B AMPLITUDE		0-16	00-10
R10	CHANNEL C AMPLITUDE		0-16	00-10
R11	ENVELOPE PERIOD	FINE TUNE ENVELOPE	0-255	00-FF
R12		COARSE TUNE ENVELOPE	0-255	00-FF
R13	ENVELOPE SHAPE/CYCLE	CONT:ATTACK:ALT: HOLD	0-15	00-0F
R14	NOT USED	REGISTER VALUE NOT SIGNIFICANT		
R15	NOT USED	REGISTER VALUE NOT SIGNIFICANT		



## 10. APPENDIX C NOISE AND TONE REGISTER

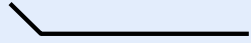
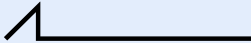








Adapted from General Instruments Programmable Sound Generator Data Manual.

REGISTER VALUE		NOISE CHANNEL			TONE CHANNEL			REGISTER VALUE		NOISE CHANNEL			TONE CHANNEL		
DEC	HEX	C	B	A	C	B	A	DEC	HEX	C	B	A	C	B	A
00	00	O	O	O	O	O	O	32	20	-	O	O	O	O	O
		N	N	N	N	N	N				N	N	N	N	N
01	01	O	O	O	O	O	-	33	21	-	O	O	O	O	-
		N	N	N	N	N					N	N	N	N	
02	02	O	O	O	O	-	O	34	22	-	O	O	O	-	O
		N	N	N	N		N				N	N	N		N
03	03	O	O	O	O	-	-	35	23	-	O	O	O	-	-
		N	N	N	N						N	N	N		
04	04	O	O	O	-	O	O	36	24	-	O	O	-	O	O
		N	N	N		N	N				N	N		N	N
05	05	O	O	O	-	O	-	37	25	-	O	O	-	O	-
		N	N	N		N					N	N		N	
06	06	O	O	O	-	-	O	38	26	-	O	O	-	-	O
		N	N	N			N				N	N			N
07	07	O	O	O	-	-	-	39	27	-	O	O	-	-	-
		N	N	N							N	N			
08	08	O	O	-	O	O	O	40	28	-	O	-	O	O	O
		N	N		N	N	N				N		N	N	N
09	09	O	O	-	O	O	O	41	29	-	O	-	O	O	-
		N	N		N	N	N				N		N	N	
10	0A	O	O	-	O	O	-	42	2A	-	O	-	O	-	O
		N	N		N	N					N		N		N
11	0B	O	O	-	O	-	-	43	2B	-	O	-	O	-	-
		N	N		N						N		N		
12	0C	O	O	-	-	O	O	44	2C	-	O	-	-	O	O
		N	N			N	N				N			N	N
13	0D	O	O	-	-	O	-	45	2D	-	O	-	-	O	-
		N	N			N					N			N	
14	0E	O	O	-	-	-	O	46	2E	-	O	-	-	-	O
		N	N				N				N				N
15	0F	O	O	-	-	-	-	47	2F	-	O	-	-	-	-
		N	N								N				
16	10	O	-	O	O	O	O	48	30	-	-	O	O	O	O
		N		N	N	N	N					N	N	N	N
17	11	O	-	O	O	O	-	49	31	-	-	O	O	O	-
		N		N	N	N						N	N	N	
18	12	O	-	O	O	-	O	50	32	-	-	O	O	-	O
		N		N	N		N					N	N		N
19	13	O	-	O	O	-	-	51	33	-	-	O	O	-	-
		N		N	N							N	N		
20	14	O	-	O	-	O	O	52	34	-	-	O	-	O	O
		N		N		N	N					N		N	N

21	15	ON	-	ON	-	ON	-	53	35	-	-	ON	-	ON	-
22	16	ON	-	ON	-	-	ON	54	36	-	-	ON	-	-	ON
23	17	ON	-	ON	-	-	-	55	37	-	-	ON	-	-	-
24	18	ON	-	-	ON	ON	ON	56	38	-	-	-	ON	ON	ON
25	19	ON	-	-	ON	ON	-	57	39	-	-	-	ON	ON	ON
26	1A	ON	-	-	ON	-	ON	58	3A	-	-	-	ON	-	ON
27	1B	ON	-	-	ON	-	-	59	3B	-	-	-	ON	-	-
28	1C	ON	-	-	-	ON	ON	60	3C	-	-	-	-	ON	ON
29	1D	ON	-	-	-	ON	-	61	3D	-	-	-	-	ON	-
30	1E	ON	-	-	-	-	ON	62	3E	-	-	-	-	-	ON
31	1F	ON	-	-	-	-	-	63	3F	-	-	-	-	-	-

## 11. APPENDIX D ENVELOPE SHAPE REGISTER

Adapted from General Instruments Programmable Sound Generator Data Manual.

REGISTER VALUE		BIT VALUES				GRAPHIC REPRESENTATION
DEC	HEX	CONT	ATTK	ALT	HOLD	
00	00	OFF	OFF	–	–	
04	04	OFF	ON	–	–	
08	08	ON	OFF	OFF	OFF	
09	09	ON	OFF	OFF	ON	
10	0A	ON	OFF	ON	OFF	
11	0B	ON	OFF	ON	ON	
12	0C	ON	ON	OFF	OFF	
13	0D	ON	ON	OFF	ON	
14	0E	ON	ON	ON	OFF	
15	0F	ON	ON	ON	ON	

## 12. APPENDIX E EQUAL TEMPERED CHROMATIC SCALE

Adapted from General Instruments Programmable Sound Generator Data Manual.

FCLOCK – 1.023Mhz			TONE PERIOD			
NOTE	OCTV	NOTE FREQ	DECIMAL		HEX	
			CRSE	FINE	CRSE	FINE
C	1	32.703	7	163	07	A3
C#	1	34.648	7	53	07	35
D	1	36.708	6	205	06	CD
D#	1	38.891	6	108	06	6C
E	1	41.203	6	15	06	0F
F	1	43.654	5	184	05	B8
F#	1	46.249	5	102	05	66
G	1	48.999	5	24	05	18
G#	1	51.913	4	207	04	CF
A	1	55.000	4	138	04	8A
A#	1	58.270	4	73	04	49
B	1	61.735	4	11	04	0B
C	2	65.406	3	209	03	D1
C#	2	69.296	3	154	03	9A
D	2	73.416	3	102	03	66
D#	2	77.782	3	54	03	36
E	2	82.406	3	7	03	07
F	2	87.308	2	220	02	DC
F#	2	92.498	2	179	02	B3
G	2	97.998	2	140	02	8C
G#	3	103.826	2	103	02	67
A	3	110.000	2	69	02	45
A#	3	116.540	2	36	02	24
B	3	123.470	2	5	02	05
C	3	130.812	1	232	01	E8
C#	3	138.592	1	205	01	CD
D	3	146.832	1	179	01	B3
D#	3	155.564	1	155	01	9B
E	3	164.812	1	131	01	83
F	3	174.616	1	110	01	6E
F#	3	184.996	1	89	01	59
G	3	195.996	1	70	01	46
G#	3	207.652	1	51	01	33
A	3	220.000	1	34	01	22
A#	3	233.080	1	18	01	12

FCLOCK – 1.023Mhz			TONE PERIOD			
NOTE	OCTV	NOTE FREQ	DECIMAL		HEX	
			CRSE	FINE	CRSE	FINE
B	3	246.940	1	2	01	02
C	4	261.624	0	244	00	F4
C#	4	277.184	0	230	00	E6
D	4	293.664	0	217	00	D9
D#	4	311.128	0	205	00	CD
E	4	329.624	0	193	00	C1
F	4	349.232	0	183	00	B7
F#	4	369.992	0	172	00	AC
G	4	391.992	0	163	00	A3
G#	4	415.304	0	153	00	99
A	4	440.000	0	145	00	91
A#	4	466.160	0	137	00	89
B	4	493.880	0	129	00	81
C	5	523.248	0	122	00	7A
C#	5	554.368	0	115	00	73
D	5	587.328	0	108	00	6C
D#	5	622.256	0	102	00	66
E	5	659.248	0	96	00	60
F	5	698.464	0	91	00	58
F#	5	739.984	0	86	00	56
G	5	783.984	0	81	00	51
G#	5	830.608	0	76	00	4C
A	5	880.000	0	72	00	48
A#	5	932.320	0	68	00	44
B	5	987.760	0	64	00	40
C	6	1046.496	0	61	00	3D
C#	6	1108.736	0	57	00	39
D	6	1174.656	0	54	00	36
D#	6	1244.512	0	51	00	33
E	6	1318.496	0	48	00	30
F	6	1396.928	0	45	00	2D
F#	6	1479.968	0	43	00	2B
G	6	1567.968	0	40	00	2B
G#	6	1661.216	0	38	00	26
A	6	1760.000	0	36	00	24
A#	6	1864.640	0	34	00	22
B	6	1975.520	0	32	00	20
C	7	2092.992	0	30	00	1E
C#	7	2217.472	0	28	00	1C
D	7	2349.312	0	27	00	1B

FCLOCK – 1.023Mhz			TONE PERIOD			
NOTE	OCTV	NOTE FREQ	DECIMAL		HEX	
			CRSE	FINE	CRSE	FINE
D#	7	2489.024	0	25	00	19
E	7	2636.992	0	24	00	18
F	7	2793.856	0	22	00	16
F#	7	2959.936	0	21	00	15
G	7	3135.936	0	20	00	14
G#	7	3322.432	0	19	00	13
A	7	3520.000	0	18	00	12
A#	7	3729.280	0	17	00	11
B	7	3951.040	0	16	00	10
C	8	4185.984	0	15	00	0F
C#	8	4434.944	0	14	00	0E
D	8	4698.624	0	13	00	0D
D#	8	4978.048	0	12	00	0C
E	8	5273.984	0	12	00	0C
F	8	5587.712	0	11	00	0B
F#	8	5919.872	0	10	00	0A
G	8	6271.872	0	10	00	0A
G#	8	6644.864	0	9	00	09
A	8	7040.000	0	9	00	09
A#	8	7458.560	0	8	00	08
B	8	7902.080	0	8	00	08

## 13. APPENDIX F ASSEMBLY LANGUAGE PROGRAM LISTINGS

### PRIMARY ROUTINES FOR SLOT 4

```

1  *PRIMARY ROUTINES
2  *FOR SLOT 4
3  *
4  ORG          $9000
5  ;ADDRESSES FOR FIRST 6522
6  ORB          EQU    $C400 ;PORT B
7  ORA          EQU    $C401 ;PORT A
8  DDRB         EQU    $C402 ;DATA DIRECTION REGISTER (A)
9  DDRA         EQU    $C403 ;DATA DIRECTION REGISTER (B)
10 ;ADDRESSES FOR SECOND 6522
11 ORB2         EQU    $C480 ;PORT B
12 ORA2         EQU    $C481 ;PORT A
13 DDRB2        EQU    $C482 ;DATA DIRECTION REGISTER (B)
14 DDRA2        EQU    $C483 ;DATA DIRECTION REGISTER (A)
15 *
16 *ROUTINES FOR FIRST 6522
17 *
18 INIT          LDA    #$FF      ;SET PORT A FOR OUTPUT 9000: A9    FF
19 STA          DDRA          9002: 8D    03    C4
20 LDA          #$07          9005: A9    07
21 STA          DDRB          ;SET PORT B FOR OUTPUT 9007: 8D    02    C4
22 RTS          ;RETURN          900A: 60
23 *
24 LATCH         LDA    #$07      ;SEND "LATCH COMMAND" 900B: A9    07
25 STA          ORB          ;TO SOUND CHIP 900D: 8D    00    C4
26 LDA          #$04          ;THROUGH PORT B 9010: A9    04
27 STA          ORB          9012: 8D    00    C4
28 RTS          ;RETURN          9015: 60
29 *
30 WRITE         LDA    #$06      ;SEND "WRITECOMMAND" 9016: A9    06
31 STA          ORB          ;TO SOUND CHIP 9018: 8D    00    C4
32 LDA          #$04          ;THROUGH PORT 901B: A9    04
33 STA          ORB          901D: 8D    00    C4
34 RTS          ;RETURN          9020: 60
35 *
36 RESET         LDA    #$00      ;SEND "RESET COMMAND" 9021: A9    00
37 STA          ORB          ;TO SOUND CHIP 9023: 8D    00    C4
38 LDA          #$04          ;THROUGH PORT B 9026: A9    04
39 STA          ORB          9028: 8D    00    C4
40 RTS          ;RETURN          902B: 60
41 *
42 *ROUTINES FOR SECOND 6522
43 *
44 INIT2         LDA    #$FF      ;SET PORT A FOR OUTPUT 902C: A9    FF
45 STA          DDRA2        902E: 8D    83    C4
46 LDA          #$07          ;SET PORT B FOR OUTPUT 9031: A9    07
47 STA          DDRB2        9033: 8D    82    C4
48 RTS          ;RETURN          9036: 60
49 *
50 LATCH2        LDA    #$07      ;SEND "LATCH COMMAND" 9037: A9    07
51 STA          ORB2         ;TO SOUND CHIP 9039: 8D    80    C4
52 LDA          #$04          ;THROUGH PORT B 903C: A9    04
53 STA          ORB2         903E: 8D    80    C4
54 RTS          ;RETURN          9041: 60
55 *
56 WRITE2        LDA    #$06      ;SEND "WRITE COMMAND" 9042: A9    06
57 STA          ORB2         ;TO SOUND CHIP 9044: 8D    80    C4
58 LDA          #$04          ;THROUGH PORT B 9047: A9    04
59 STA          ORB2         9049: 8D    80    C4
60 RTS          ;RETURN          904C: 60
61 *
62 STA          #$00          ;SEND "RESET COMMAND" 904D: 9A    00
63 STA          ORB2         ;TO SOUND CHIP 904F: 8D    80    C4
64 LDA          #$04          ;THROUGH PORT B 9052: A9    04
65 STA          ORB2         9054: 8D    80    C4
66 RTS          ;RETURN          9057: 60

```

## TABLE ACCESS ROUTINE FOR SLOT 4

```

1  "TABLE ACCESS ROUTINE
2  "FOR SLOT 4
3  *
4  ORG      $8000
5  ;ADDRESSES FOR FIRST 6522
6  PTR      EQU      $08      ;DATA POINTER
7  ORA      EQU      $C401    ;PORT A
8  LATCH    EQU      $900B    ;LATCH SUB-ROUTINE
9  WRITE    EQU      $9016    ;WRITE SUB-ROUTINE
10 RESET    EQU      $9021    ;RESET SUB-ROUTINE
11 ;ADDRESSES FOR SECOND 6522
12 PRT2     EQU      $0A      ;DATA POINTER
13 ORA2     EQU      $C481    ;PORT A
14 LATCH2   EQU      $9037    ;LATCH SUB-ROUTINE
15 WRITE2   EQU      $9042    ;RESET SUB-ROUTINE
16 RESET2   EQU      $904D    ;RESET SUB-ROUTINE
17 *
18 *ROUTINES FOR FIRST 6255
19 *
20 START     JSR RESET        ;RESET SOUND CHIP      8000: 20      21      90
21 LDY       #$00            ;USED TO IDENTIFY REGISTER 8003: A0      00
22 LOOP      STY      ORA      ;# OF SOUND CHIP      8005: 8C      01      C4
23 JSR       LATCH          ;                        8008: 20      0B      90
24 LDA       (PTR),Y         ;GET DATA FROM TABLE  800B: B1      08
25 STA       ORA            ;                        800D: 8D      01      C4
26 JSR       WRITE          ;STORE IN REGISTER       8010: 20      16      90
27 CPY       #$0F            ;END OF DATA?          8013: C0      0F
28 BEQ       DONE           ;YES, EXIT               8015: F0      04
29 INY
30 JMP       LOOP            ;NO, GET NEXT SET        8017: CB
31 DONE      RTS             ;RETURN                 8018: 4C      05      80
32 *
33 *ROUTINES FOR SECOND 6522
34 *
35 START2    JSR      RESET2  ;SAME INSTRUCTIONS AS  801B: 60
36 LDY       #$00            ;ABOVE                 801C: 20      4D      90
37 LOOP2     STY      ORA2    ;                        801F: A0      00
38 JSR       LATCH2         ;                        8021: 8C      81      C4
39 LDA       (PTR2),Y        ;                        8024: 20      37      90
40 STA       ORA2           ;                        8027: B1      0A
41 JSR       WRITE2         ;                        8029: 8D      81      C4
42 CPY       #$0F            ;                        802C: 20      42      90
43 BEQ       DONE2          ;                        802F: C0      0F
44 INY
45 JMP       LOOP2          ;                        8031: F0      04
46 DONE2     RTS            ;                        8033: CB
                        ;                        8034: 4C      22      80
                        ;                        8037: 60

```



## PROCESSOR LOOP SOUND EFFECT

```

1  "PROCESSOR LOOP
2  "FOR LASER AND BOMB
3  "SOUND EFFECT
4  *
5  ORG      $8F00
6  *;FOR FIRST 6522
7  *
8  PTR      EQU      $08      ;DATA POINTE
9  TONE     EQU      $0A      ;TONAL VALUE
10 TIME    EQU      $0B      ;TIME VALUE FOR DELAY
11 BASE     EQU      $C400    ;CARD ADDRESS
12 ORA      EQU      BASE+1   ;PORT A
13 TAR      EQU      $8000    ;TABLE ACCESS ROUTINE
14 LATCH    EQU      $900B    ;LATCH SUBROUTINE
15 WRITE    EQU      $9016    ;WRITE SUBROUTINE
16 RESET    EQU      $9021    ;RESET SUBROUTINE
17 WAIT     EQU      $FCAB    ;WAIT SUBROUTINE
18 *
19 *
20 LASER    LDA      #$00      ;LOAD HIGHEST      8F00:  A9      00
21 STA      TONE      ;FREQUENCY VALUE      8F02:  85      0A
22 LDA      #$0F      ;LOAD SHORT      8F04:  A9      0F
23 STA      TIME      ;TIME DELAY      8F06:  85      0B
24 JMP      START     ;AND START      8F08:  4C      13      8F
25 BOMB     LDA      #$30      ;LOAD MIDDLE      8F0B:  A9      30
26 STA      TONE      ;FREQUENCY VALUE      8F0D:  85      0A
27 LDA      #$40      ;LOAD LONGER      8F0F:  A9      40
28 STA      TIME      ;TIME DELAY      8F11:  85      0B
29 START    LDA      #$90      ;SET TABLE ADDRESS 8F13:  A9      90
30 STA      PTR      ;      8F15:  85      08
31 STA      PTR+1     ;      8F17:  A9      81
32 LDA      #$81      ;      8F19:  85      09
33 STA      PTR+1     ;      8F1B:  20      00      80
34 JSR      TAR        ;TRANSFER DATA      8F1E:  A9      00
35 LDA      #$00      ;LATCH FIRST REGISTER 8F20:  8D      01      C4
36 STA      ORA        ;ADDRESS      8F23:  20      0B      90
37 JSR      LATCH      ;      8F26:  A5      0A
38 LOOP     LDA      TONE    ;GET TONE VALUE 8F28:  8D      01      C4
39 STA      ORA        ;STORE IN REGISTER 8F2B:  20      16      90
40 JSR      WRITE      ;      8F2E:  A5      0B
41 LDA      TIME      ;GET TIME VALUE      8F30:  20      A8      FC
42 JSR      WAIT      ;AND DELAY      8F33:  E6      0A
43 INC      TONE      ;INCREMENT TONE VALUE 8F35:  A9      FF
44 LDA      #$FF      ;END OF INCREASE? 8F37:  C5      0A
45 CMP      TONE      ;      8F39:  F0      03
46 BEQ      DONE      ;YES,EXIT      8F3B:  4C      26      8F
47 JMP      LOOP      ;NO, START AGAIN      8F3E:  A5      0B
48 DONE    LDA      TIME    ;GET TIME VALUE 8F40:  20      AB      FC
49 JSR      WAIT      ;DELAY      8F43:  A9      00
50 LDA      #$00      ;RESTORE ORIGINAL      8F45:  85      0A
51 STA      TONE      ;TONE VALUE      8F47:  20      21      90
52 JSR      RESET      ;CLEAR SOUND CHIP      8F4A:  60
53 RTS        ;REGISTERS AND RETURN

```

## SPEECH COMPOSITE DRIVER

```

1  *SPEECH-COMPOSITE DRIVER
2  *
3  ORG                $6D00
4  *
5  TABPTR             EQU    $F9           ;DATA POINTER
6  OUTPTR             EQU    $FB           ;START OF DATA POINTER
7  ENDPTR             EQU    $FD           ;END OF DATA POINTER
8  BUSY               EQU    $FF           ;BUSY FLAG
9  IRQL               EQU    $03FE        ;INTERRUPT VECTOR, LOW BYTE
10 IRQH               EQU    $03FF        ;INTERRUPT VECTOR, HIGH BYTE
11 BASE               EQU    $C440        ;FIRST SPEECH CHIP
12 DURPHON            EQU    BASE         ;REGISTER 0 OF SPEECH CHIP
13 INFLECT            EQU    BASE+$01     ;REGISTER 1 OF SPEECH CHIP
14 RATEINF            EQU    BASE+$02     ;REGISTER 2 OF SPEECH CHIP
15 CTTRAMP            EQU    BASE+$03     ;REGISTER 3 OF SPEECH CHIP
16 FILFREQ            EQU    BASE+$04     ;REGISTER 4 OF SPEECH CHIP
17 DDRB               EQU    $C402
18 DDRA               EQU    $C403
19 PCR                EQU    $C0C         ;PERIPHERAL CONTROL REG-6522
20 IFR                EQU    $C40D        ;INTERRUPT FLAG REG-6522
21 PCR                EQU    $C48C        ;INTERRUPT ENABLE REG-6522
22 IFR                EQU    $C48D
23 IER                EQU    $C48E
24 *SETUP ROUTINE
25 *
26 SEI                ;DISABLE INTERRUPTS      6D00: 78
27 LDA    #<INTERR    ;SET INTERRUPT VECTOR  6D01: A9      4D
28 STA    IRQL        ;TO POINT TO INTERRUPT 6D03: 8D      FE      03
29 LDA    #>INTERR    ;SERVICE ROUTINE      6D06: A9      6D
30 STA    IRQH        ;                      6D08: 8D      FF      03
31 *
32 LDA    #$00        6D0B: AS      FA
33 STA    DDRA
34 STA    DDRB
35 TABPTR+1           ;GET HIGH ADDRESS OF DATA 6D0D: 85      FC
36 OUTPTR+1           ;STORE IN WORK POINTER    6D0F: A6      F9
37 TABPTR             ;GET LOW ADDRESS OF DATA 6D11: E8
38 INX                ;INCREMENT TWICE         6D12: E8
39 INX                ;TO SKIP OVER LENGTH BYTES 6D13: D0      02
40 BNE    CONT        ;CHECK FOR PAGE BOUNDARY 6D15: E6      FC
41 INC    OUTPTR+1    6D17: 86      FB
42 CONT    STX    OUTPTR ;STORE LOW BYTE
43 *
44 LDY    #$01        6D19: A0      01
45 LDA    (TABPTR),Y   ;GET HIGH LENGTH BYTE    6D1B: B1      F9
46 CLC                6D1D: 18
47 ADC    TABPTR+1     ;AND ADD TO BASEADDRESS  6D1E: 65      FA
48 STA    ENDPTR+1     ;STORE ENDADDRESS        6D20: 85      FE
49 DEY                6D22: 88
50 LDA    (TABPTR),Y   ;GET LOW LENGTH BYTE    6D23: 81      F9
51 CLC                6D25: 18
52 ADC    TABPTR       ;AND ADD TO BASE ADDRESS 6D26: 65      F9
53 BCC    CONT1        ;CHECK FOR PAGE BOUNDARY 6D28: 90      02
54 INC    ENDPTR+1     6D2A: E6      FE
55 CONT1    STA    ENDPTR ;STORE END ADDRESS    6D2C: 85      FD
56 *
57 CONT5    LDA    #$FF ;SET BUSY FLAG          6D2E: A9      FF
58 STA    BUSY         ;AND SET PERIPHERAL CONTROL 6D30: 85      FF
59 LDA    #$0C         ;REGISTER TO RECOGNIZE    6D32: A9      0C
60 STA    PCR          ;SIGNAL FROM SPEECH CHIP  6D34: 8D      0C      C4
61 LDA    #$80         ;RAISE CTRL BIT IN REGISTER 3 6D37: A9      80
62 STA    CTTRAMP      6D39: 8D      23      C4
63 LDA    #$C0         ;SET TRANSITIONED INFLECTION 6D3C: A9      C0
64 STA    DURPHON      6D3E: 8D      20      C4
65 LDA    #$70         ;MODE IN REGISTER 0       6D41: A9      70
66 STA    CTTRAMP      6D43: 8D      23      C4
67 LDA    #$82         ;ENABLE 6522 INTERRUPTS    6D46: A9      82
68 STA    IER          6D48: 8D      0E      C4
69 CLI                ;CLEAR INTERRUPT MASK      6D48: 58
70 RTS                ;RETURN TO CALLER          6D4C: 60
71 *INTERRUPT ROUTINE
72 INTERR    TXA        ;SAVE REGISTERS          6D4D: 8A
                          6D4E: 48

```

```








73 PHA                                6D4F: 98
74 TYA                                6D50: 48
75 PHA                                6D51: A9    02
76 LDA    #$02                      ;CLEAR INTERRUPT FLAG  6D53: 8D    0D    C4
77 STA    IFR                        6D56: A0    00
78 LDY    #$00                      ;INIT REGISTERS        6D58: A2    04
79 LDX    #$04                      6D5A: A5    FB
80 LDA    OUTPRT                    ;CHECK FOR END OF DATA FILE 6D5C: C5    FD
81 CMP    ENDPTR                    6D5E: D0    20
82 BCC    CONT6                     ;IF NOT THEN CONTINUE    6D60: A5    FC
83 LDA    OUTPRT+1                  ;CHECK HIGH ADDRESS ALSO 6D62: C5    FE
84 CMP    ENDPRT+1                 6D64: D0    1A
85 BCC    CONT6                     ;IF NOT THEN CONTINUE    6D66: A9    00
86 LDA    #$00                      ;IF END, TURN EVERY-THING OFF 6D68: 8D    20    C4
87 STA    DURPHON                  ;STORE PAUSE PHONEME    6D68: A9    70
88 LDA    #$70                      ;ZERO AMPLITUDE         6D6D: 8D    23    C4
89 STA    CTTRAMP                  6D70: A9    00
90 LDA    #$00                      ;CLEAR BUSY FLAG        6D72: 85    FF
91 STA    BUSY                     6D74: A9    02
92 LDA    #$02                      ;CLEAR INTERRUPT ENABLE 6D76: 8D    0E    C4
93 STA    IER                      ;IN 6522                6D79: 68
94 LDA    #$FF                     6D7A: A8
95 STA    DDRA                     6D7B: 68
96 LDA    #$07                     6D7C: AA
97 STA    DDRB                     6D7D: A5    45
98 RET    PLA                      ;RESTORE REGISTERS      6D7F: 40
99 TAY                             6D80: B1    FB
100 PLA                             6D82: 9D    20    C4
101 TAX                             6D8F: E6    FB
102 LDA    $45                      ;NEXT DATA            6D87: D0    02
103 RTI                             ;RETURN FROM INTERRUPT 6D89: E6    FC
104 *                               6D88: CA
105 CONT6 LDA(OUTPTR),Y             ;GET DATA             6D8C: E0    FF
106 STA    BASE,X                   ;STORE IN SPEECH CHIP  6D8E: D0    F0
107 INC    OUTPTR                   ;NEXT DATA            6D90: F0    E7
108 BNE    CONT7
109 INC    OUTPTR+1
110 *
111 CONT7 DEX                       ;NEXT REGISTER
112 CPX    #$FF                     ;LAST REGISTER?
113 BNE    CONT6                     ;NO, CONTINUE
114 BEQ    RET                       ;YES, RETURN

```

## 14. COMPATIBLE SOFTWARE




TITLE	COMMENT	TITLE	COMMENT
Adv. Construction Set		One on One	
Airsim-3		Phasor Software	Applied Engineering
Apple Cider Spider		Pitfall II	
Auto Gyro		Popeye	
Bank Street Music Writer		Rescue Raiders v1.3	Speech Only
Berzap! - Berzerk Clone		Saxophone Master	
Bouncing Kamungas		Silent Service	
Broadsides	Speech supported	Singing Master	
Crimewave		Skyfox	
Clarinet Master		Spy Strikes Back	
Crypt of Medea	Speech supported	Tactical Armor Command	
Cybernoid Music Disk		Thunder Bombs	
Flute Master		Trumpet Master	
Guitar Master	Guitar tutoring	Ultima III Exodus	Specific version
Gl Joe		Ultima IV	
Lady Tut	Specific version	Ultima V	Supports two boards
Lancaster		Under Fire	
Maze Craze		Willy Byte	
Developers Kit	Sweet Micro Systems	Window	Window
Speech Developers Kit	Sweet Micro Systems	Zaxxon	Specific version
Music Construction Set	Revisions do more	ZooKeeper	
Music Star	Patched version	One on One	
Night Flight			

## 15. MOCKINGBOARD REVISIONS

      				
Release Date	Revision	Expansion Slot Required	Kit Assembly*	Ready to use
2005	1.0	YES	YES	YES
2010	1.1	YES	YES	YES
2017	2.1	YES	YES	YES
2018	2.2	YES	YES	YES
2018-08-14	2.3	YES	YES	YES

\*Kit assembly is available for do it yourself hobbyists, for those who have experience in building an expansion card using a solder gun and a basic understanding of electronics, if you are unsure, the factory made ready to use expansion card is available.

Enquires can be made at: [support@reactivemicro.com](mailto:support@reactivemicro.com)

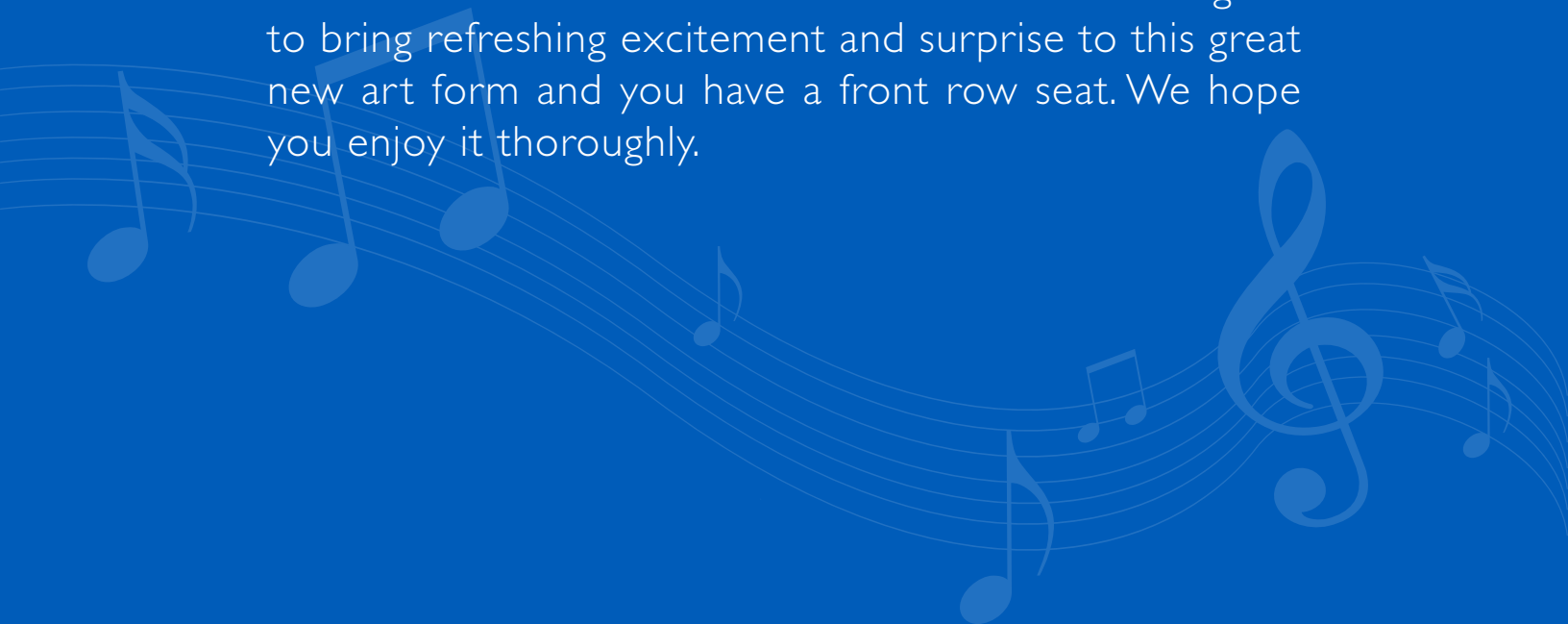
  			
Release Date	Revision	Internal Installation only	Some assembly required
2018	1.0	YES	YES
2021	1.0B	YES	YES

Enquires can be made at: [quick09@gmail.com](mailto:quick09@gmail.com)

Sweet Micro Systems	
Model	Comments
A	Two AY-3-8913 chips for six audio channels and two open sockets for SSI-263 speech chips.
B	SSI-263 speech chip upgrade for Mockingboard A.
C	Two AY-3-8913 and one SSI-263 (essentially a Mockingboard A with the upgrade pre-installed, only one speech chip allowed).
D	Apple IIc only, two AY-3-8913 and one SSI-263.
M	Bundled with Mindscape's Bank Street Music Writer. Came with two AY-3-8913 chips and an open socket for one speech chip. This model included a headphone jack and a jumper to permit sound to be played through the Apple Computer's built-in speaker.

# mockingboard

Fine Art pleases man as much by what it implies as by what it depicts. Theater succeeds equally by the actual and the illusory. But, in certain human activities, what is not supplied detracts. We have come to expect from computers fine graphics and animation, thoughtful adventures and new approaches to education. Yet, in the silent delivery of these marvels something is missing. Sound! MOCKINGBOARD was developed to give the programmer and the player, the teacher and the pupil, computer applications loaded with vitality, music, speech and sound effects. With MOCKINGBOARD, the computer moves from silence to stereo and using the computer will never be the same. MOCKINGBOARD was designed to bring refreshing excitement and surprise to this great new art form and you have a front row seat. We hope you enjoy it thoroughly.





[khaibitgfx@gmail.com](mailto:khaibitgfx@gmail.com)